MEJORAMIENTO DE PROGRAMACIÓN DE PRODUCCIÓN EN PLANTA DE INYECCIÓN DE PLÁSTICOS USANDO UN ALGORITMO GENÉTICO

AUTOR:

OSCAR HERNANDO ESTUPIÑÁN BELTRÁN

UNIVERSIDAD DE LA SABANA

FACULTAD DE INGENIERÍA

MAESTRÍA EN GERENCIA DE INGENIERÍA

CHÍA

2021

MEJORAMIENTO DE PROGRAMACIÓN DE PRODUCCIÓN EN PLANTA DE INYECCIÓN DE PLÁSTICOS USANDO UN ALGORITMO GENÉTICO

AUTOR:

OSCAR HERNANDO ESTUPIÑÁN BELTRÁN

Proyecto de grado elaborado como requisito para optar al título de Magister en Gerencia de Ingeniería.

DIRECTOR DE PROYECTO:

DAVID L. CORTÉS-MURCIA, PhD.

UNIVERSIDAD DE LA SABANA

FACULTAD DE INGENIERÍA

MAESTRÍA EN GERENCIA DE INGENIERÍA

CHÍA

2021

Nota de aceptación:

_____
_____
_____
_____
_____

_____
Firma de presidente de jurado

_____
Firma de jurado

_____
Firma de jurado

Chía, 15/Junio/2021

Dedicatoria.

A mi Mamá, quien con su determinación, amor e incondicionalidad me ha apoyado en todos los proyectos que he emprendido, le debo todo.

Agradecimientos.

Agradezco a la Universidad de la Sabana por propiciar un entorno adecuado para desarrollar la maestría.

Agradezco totalmente al Profesor David L. Cortés Murcia, su gran profesionalismo, rigurosidad, experiencia e interés en el proyecto fueron determinantes para desarrollarlo exitosamente. También agradezco todas sus lecciones, complementaron mi formación como magister.

CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

Este proyecto de grado buscó solucionar el problema de secuenciar un conjunto de trabajos y moldes de inyección en máquinas inyectoras en una planta de inyección, con el fin de reducir el makespan y tardanza. La planta de inyección es el primer eslabón de la cadena de producción en una fábrica de productos de consumo masivo. El proyecto se caracterizó como un problema de secuenciación de trabajos en máquinas no relacionadas paralelas. El proyecto aborda el problema con dos métodos, el primero usando programación lineal entera mixta (MILP) y el segundo usando un algoritmo genético. El método exacto funciona bien con instancias pequeñas de máximo 10 trabajos y cinco máquinas. Respecto al segundo método se diseñó un algoritmo genético con una función fitness completamente original, el algoritmo genético permite encontrar soluciones de calidad en corto tiempo para instancias más grandes y complejas si se compara con el método exacto. El algoritmo genético requirió un ajuste de sus parámetros usando diseño factorial multinivel. Con el objetivo de probar el método exacto de solución y lograr una comparación estricta entre los dos métodos de solución se desarrollaron instancias de dos tipos: random y reales con información de la planta de inyección. Luego se desarrollaron experimentos computacionales solucionando las instancias con los dos métodos. Los resultados de los experimentos permitieron establecer que el algoritmo genético propuesto genera soluciones iguales o mejores en instancias random comparado con el método exacto. Se desarrolló un caso de estudio que representa la programación de producción en un mes para piezas tipo A, este caso es usado para comparar el método de programación usado actualmente en la planta de inyección versus el algoritmo genético propuesto. El algoritmo genético, crea un programa de secuenciación de trabajos con un tiempo máximo de terminación y tardanza 30% menores que el método actual.

ABSTRACT

This thesis developed a method to find solutions to a scheduling problem in an injection mold factory, intending to reduce the makespan and tardiness. The injection mould factory is the first stage in a massive consumer product factory location. The project was characterized as a nonrelated parallel machine scheduling problem. The project approached the problem with two methods: the 1st one using mixed-integer linear programming (MILP) and the 2nd one using a genetic algorithm. The exact solution method works fine with small instances, with a maximal size of ten jobs and five machines. About the second method, a genetic algorithm was designed with a completely original fitness function, the genetic algorithm was able to find several quality solutions in a shorter time for larger and complex instances if it is compared with the exact solution method. The genetic algorithm required several parameter adjustments, the multilevel factorial design was used to do so. With the objective of testing the exact method and to can achieve a strict comparison between both solution methods, two kinds of instances were developed: 1 kind with random data and the other one with real production data. After it, several computational experiments were developed, solving the whole instances with both solution methods. Experiments results allowed the researchers to conclude that the proposed genetic algorithm creates equal or better solutions if it is compared with the exact solution method with random instances. One study case was developed, this case represents the production schedule for a month for injected components labeled as type A, the study case was used to compare the current scheduling method used in the injection factory versus the proposed genetic algorithm. The genetic algorithm provides a scheduling program with Cmax and Tardiness values 30% lower than the current scheduling method.

1. INTRODUCCION

1.1. PLANTEAMIENTO Y JUSTIFICACIÓN DE LA INVESTIGACIÓN

La industria de transformación de plástico por moldeo de inyección representa un 15% (Acoplasticos, 2018) del GDP (Gross Domestic Product, Producto Interno Bruto). El mercado de transformación de polímeros está directamente relacionado con los principales commodities, como el petróleo (del cual se derivan), la construcción y el sector automovilístico (Alcaldía Medellín-CREAME, 2019). La industria de inyección de plástico es uno de los proveedores más importantes de semielaborados para la fabricación de productos de consumo masivo, eje fundamental de la economía (mincit, 2019).

La industria de inyección de plástico también soporta otros sectores de la economía de forma indirecta, esta relación se observa especialmente en la industria metalmecánica porque la transformación del plástico requiere de un brochure extenso de servicios tercerizados para la fabricación y mantenimiento de los moldes de inyección y maquinaria, incluyendo todos sus periféricos. A nivel nacional la industria está bien posicionada y gran parte de los componentes necesarios por el mercado local se producen internamente, además Colombia se ha convertido en un punto estratégico para abastecer de componentes inyectados a varios países de Latinoamérica compitiendo directamente con Brasil y México (mincit, 2019).

El principio de moldeo por inyección es básico pero el proceso de secuenciar los trabajos en las plantas de inyección es complejo por la cantidad de restricciones presentes relacionadas con compatibilidad recursos para realizar una tarea, incluyendo su respectiva disponibilidad y duplicidad. Además, las plantas de inyección son en muchas ocasiones el primer eslabón en la cadena de suministro en clusters de manufactura, exigiendo al máximo que se cumplan los cronogramas de producción presupuestados.

Este proyecto de grado está inspirado por los retos planteados anteriormente, se identificó la necesidad de proponer un método para programar los moldes y trabajos en una planta de inyección nacional para cumplir con los tiempos de entrega, objetivo que no se cumplen con el método actual. El proyecto también está inspirado en la oportunidad de proponer un método para secuenciar tareas y los recursos para desarrollarlas, representados en los moldes de inyección, este tipo de secuenciación no se observa en la literatura. La planta de inyección tiene un ambiente de producción complejo, existen 40 máquinas y 200 moldes operativos, algunos con réplicas y con menores tiempos de inyección que los moldes originales.

Cuando se programa la producción se debe procurar la utilización al máximo de las máquinas y moldes disponibles, buscando la oportunidad de usar los más eficientes, es decir moldes con réplicas, actualmente la compañía no aprovecha estos recursos. La producción anual es de 20.000.000 de piezas, con diferentes tamaños, materiales y tiempos de inyección.

El método actual para secuenciar los trabajos en la planta consiste en desagregar la producción, validar disponibilidad de recursos y realizar la menor cantidad de cambios en recursos y máquinas, luego se programa usando como referencia un gráfico de Gantt, proceso netamente manual, sujeto a fallos y lento.

La gestión de estrategias para cumplir los programas de producción desde una perspectiva académica o industrial se enmarcan en un campo del conocimiento denominado formalmente *problemas de secuenciación*. La relación entre industria y academia para solucionar los problemas de secuenciación ha evolucionado con los años con un intercambio continuo de información, aportando un beneficio mutuo y continuo.

Existe un gran compendio de problemas de secuenciación, justificado en su relación directa con los diferentes sistemas de producción, en la literatura consultada para desarrollar este proyecto se observó que es común el estudio de la secuenciación de trabajos en máquinas paralelas, pero los estudios como este proyecto enfocado en *máquinas no relacionadas paralelas* es reducido.

Este entorno de máquinas no relacionadas paralelas, más conocido como Rm, representa apropiadamente los entornos reales de producción acorde a la investigación de Dastidar et al. (2005). Los autores afirman que se han desarrollado exhaustivos proyectos para mejorar la programación de plantas de inyección con el objetivo de cumplir cronogramas de producción, pero aún quedan muchos problemas por resolver, porque cada planta de inyección tiene su ambiente de producción particular.

Tal es el caso de este proyecto que se fundamentó en el estado del arte y evaluó las condiciones particulares de la planta de inyección con información de piso para plantear de forma integral un método alternativo pero eficiente para secuenciar los trabajos, teniendo en cuenta las restricciones puntuales como réplicas de moldes, elegibilidad de máquina, tiempos de alistamiento y fechas de entrega.

El problema de secuenciación de trabajos para la planta de inyección se modeló como un MILP (problema lineal entero mixto) y se propuso un algoritmo genético como método de solución. Un conjunto de instancias de dos tipos fueron

formuladas: random y reales con información de producción. El conjunto de instancias fue resuelto con los diferentes métodos de solución para probar su desempeño.

1.2. PREGUNTA DE INVESTIGACIÓN

La dirección de la planta de plásticos tiene como indicadores prioritarios el cumplimiento del programa de producción en tiempo, cantidad y calidad, además busca aprovechar al máximo la disponibilidad de máquina, estos indicadores están enmarcados en la minimización del makespan. La dirección también tiene indicadores de nivel de servicio que buscan minimizar la tardanza en la entrega de los lotes de producción. La dirección procura un balance entre el performance de la planta y el nivel de servicio a otras áreas, por tal razón se formula la siguiente pregunta de investigación anidada que aborda el problema desde una perspectiva académica para validar los métodos de solución y aplicándolos posteriormente a un caso real.

¿Cómo se verán afectados los indicadores de makespan y tardiness para el programa de producción de la planta de inyección si se optimiza la programación con un modelo matemático de secuenciación de máquinas no relacionadas paralelas con restricciones de alistamiento, tiempos de entrega y elegibilidad de máquina?

¿Cómo se puede resolver el modelo matemático planteado para obtener un programa de secuenciación de moldes de buena calidad y en tiempo razonable?

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Diseñar un programa de producción a partir de la solución de un modelo de secuenciación de tareas para observar cómo se desempeñan los indicadores de la planta de inyección representados en el makespan y tardiness.

1.3.2. OBJETIVOS ESPECIFICOS

- Modelar a través de un MILP el problema de la secuenciación de trabajos y moldes.
- Solucionar y validar el modelo matemático con herramientas de optimización.
- Proponer un método heurístico para encontrar soluciones de alta calidad para el problema, en tiempo razonable.
- Comparar el método actual de secuenciación con el método propuesto, experimentado con la información disponible.

1.4. ESTRUCTURA DEL PROYECTO DE GRADO

En el Capítulo 2 se desarrolló una revisión general de la literatura incluyendo tendencias actuales en secuenciación de trabajos. En el Capítulo 3 se formuló el modelo MILP y se realizó una explicación detallada del conjunto de restricciones. En el Capítulo 4 se presenta el algoritmo genético como propuesta de solución y se explica cada uno de sus componentes con un ejemplo numérico. En el capítulo 5 están los experimentos computacionales realizados, se describe el proceso de elaboración de las instancias y se presentan los resultados de cada método de solución, también se describe el método para ajustar los parámetros del algoritmo genético propuesto, la descripción extendida del método de ajuste está en el Anexo A. Por último, en el Capítulo 6 se presentan las conclusiones del proyecto y el trabajo futuro a realizar.

2. ESTADO DEL ARTE

2.1. SECUENCIACIÓN DE TRABAJOS.

La secuenciación de actividades consiste en asignar recursos limitados para realizar trabajos en unos intervalos de tiempo, con el objetivo de optimizar el rendimiento del sistema. La secuenciación de trabajos surgió como una necesidad a nivel industrial para mantener la competitividad de las empresas. La programación de los trabajos está restringida por limitantes de tiempo, recursos, distancias, fechas de entrega, entre otras, que buscan simular los escenarios a nivel industrial.

Gantt (1903) fue un pionero en las secuenciaciones de trabajo con sus trabajos a inicios del siglo XX. Luego, la investigación en secuenciación estuvo ralentizada por un tiempo prolongado, hasta los años 1950, siendo el matemático Richard Bellman quien acoto por primera vez el término “scheduling” en su investigación sobre teoría de secuenciación (Bellman, 1955).

En la década de 1950 se publicaron varias investigaciones por la Universidad de California enfocadas en optimizar líneas de producción (James, 1955). La notación usada en ese momento para identificar trabajos, máquinas, duraciones y fechas, son el estándar que se mantiene hasta hoy. Pinedo (2016) representa en su trabajo este estándar de forma holística facilitando un uso común de la nomenclatura entre los diferentes autores, en la Sección 2.1.1. se realiza una breve compilación de la nomenclatura que se utiliza en este proyecto.

En las siguientes dos décadas, los estudios de teoría de secuenciación de trabajos estuvieron orientadas a los avances en programación dinámica y programación entera. En la década de 1970 la investigación estuvo enfocada al desarrollo de complejos algoritmos y programación estocástica. La publicación de Karp (1972), sobre la complejidad de computaciones computarizadas, fue un hito para el avance de la teoría de secuenciación y un referente vigente hasta hoy. Los autores Tanaev et al. (2012) afirman que en la década de 1980 también se desarrollaron avances en la teoría de secuenciación, especialmente en sistemas de múltiples etapas y relacionados con la investigación de Karp (1972), quien también definió de forma sistemática los alcances de la secuenciación de trabajos.

La investigación en problemas de secuenciación abarca modelos determinísticos y estocásticos, desde problemas sencillos donde se asigna una serie de trabajos a una máquina hasta modelos estocásticos y de incertidumbre donde no se conocen

con antelación los valores reales de las capacidades de los recursos y los tiempos de trabajo de las actividades a desarrollar.

Este proyecto está enfocado en modelos determinísticos, porque esta soportado por la información almacenada por años en la planta de plásticos referente a número de moldes, cavidades, máquinas de inyección, además de tiempos de proceso y alistamiento respectivamente.

2.1.1. NOMENCLATURA Y MODELOS DE SECUENCIACIÓN.

El autor Pinedo (2016) propone basado en la literatura una nomenclatura y tipos de modelos de secuenciación, como se muestra a continuación.

Los elementos básicos de un problema de scheduling son:

n = número de trabajos.

m = número de máquinas.

$Indice\ j$ = trabajo.

$Indice\ i$ = máquina.

De esta forma el conjunto (i, j) hace referencia a que el trabajo j se debe realizar en la máquina i.

pij = tiempo de procesamiento del trabajo j en la máquina i.

dj = fecha de cumplimiento del trabajo j.

Conociendo la notación estándar, se debe identificar el “ambiente máquina”, para nuestro proyecto nos interesa el siguiente:

Máquinas no relacionadas en paralelo ($\boldsymbol{Rm}$): están disponibles múltiples máquinas en paralelo, donde cualquier trabajo j se puede realizar, los tiempos dependen del recurso usado para realizar el trabajo. El trabajo puede estar restringido a un conjunto de máquinas idénticas, entonces se añade la restricción Mj.

Mj = restricción de elegibilidad de máquina.

También es necesario definir las siguientes medidas de desempeño del problema:

Cj = tiempo de terminación del trabajo j.

$Tj = max\ (Cj - dj)$ = el valor de la tardanza.

$Cmax = makespan$, es el tiempo de terminación del último trabajo asignado, o en otros términos $max(C_1, \dots, C_n)$.

2.2. TENDENCIA ACTUAL EN SOLUCIÓN DE PROBLEMAS DE SECUENCIACIÓN.

Gran parte de los estudios realizados en la última década, muestran un enfoque a minimizar el makespan acorde a Yang (2009). Además, se observa que en gran parte de las investigaciones no se considera el Set Up time de los procesos productivos, es una tendencia en diferentes investigaciones que se asuma el set up time en los tiempos de proceso.

Este tipo de parametrización no refleja la realidad de un proceso industrial, porque durante el set up de las máquinas y herramientas surgen muchas novedades, si se considera el Set Up time en el tiempo de proceso se está sesgando el modelo y se idealiza su comportamiento. Por otra parte, los tiempos de Set Up son un componente critico de los problemas de secuenciación porque son sensibles a la secuencia de las tareas, dependiendo de la tarea predecesora toman un valor u otro.
En su trabajo de investigación los autores Zaidi et al. (2010) crearon una metaheurística hibrida que minimiza el flujo de tiempo o tiempo ponderado de terminación. Esta metaheurística usa planteamientos basados en el estado del arte, pero innova creando nuevas funciones objetivo o incluyendo restricciones de los recursos que no se planteaban previamente.

Hasta ese año no existía un algoritmo para solucionar este tipo de problema en tiempo polinómico, por lo tanto, los autores crearon una combinación del algoritmo de búsqueda variable en vecindario (VNS), algoritmo genético (GA) y algoritmo diferencial evolucionario (DE), con el fin de reducir los costos los autores usaron solo dos vecindarios. Sus experimentos permitieron concluir que de todas las combinaciones posibles la mejor es la de los algoritmos VNS y DE.

Un caso aplicado a la industria de inyección de plásticos fue desarrollado por Wang et al. (2015). Este trabajo permitió definir una función multiobjetivo para secuenciación de máquinas idénticas paralelas, el trabajo es innovador porque incluye restricciones de mantenimiento preventivo de las recursos y minimización del makespan.

En el trabajo de Vincent et al. (2016) se desarrolló un método para minimizar la tardanza, basado en la heurística de recocido simulado, su mayor aporte es que se encontró una solución óptima para el problema de secuenciación de máquinas idénticas en paralelo. Los autores afirman que el éxito de su método se da por el uso de reglas de dominio que limitan el número de trabajos a secuenciar simplemente a los necesarios.

Un aporte importante, que no se había trabajado anteriormente, fue el uso de restricciones de retención (preempt), el modelo de Soper et al. (2019) secuencia trabajos en tres máquinas paralelas idénticas para reducir el makespan, los autores usaron un algoritmo de retención de una sola unidad (Q2Pr1), logrando la solución del problema en tiempo polinómico.

Un caso holístico como el de Afzalirad et al. (2016), corresponde a la secuenciación de máquinas no paralelas, en el que de forma integral se usan modelos MILP y algoritmos genéticos como método de solución para reducir el makespan en una fábrica de barcos. La complejidad de la operación requiere un método sofisticado como el algoritmo genético, pero surgen problemas para encontrar soluciones factibles cuando se hacen crossovers, problema común en estos algoritmos, que los autores solventan usando heurísticas correctivas. Además, se enfatiza en la necesidad de realizar buenos ajustes de parámetros.

En ocasiones se usan métodos exactos o de programación entera como el propuesto por Adulyasak et al. (2019). En esta investigación se combinaron heurísticas para solucionar los problemas por etapas. Los autores secuenciaron los trabajos en una industria de papel, dimensionando lotes y secuenciando las actividades, es extraño encontrar soluciones exactas, estas se lograron con un algoritmo del tipo Branch and Check.

Los algoritmos del tipo poblacional son usados comúnmente para solucionar los problemas más complejos de secuenciación de máquinas paralelas, en un estudio realizado en la industria de inyección realizado por Oztemel et al. (2017) se propone un modelo para secuenciar trabajos restringidos por múltiples modos y recursos. Ellos usaron un algoritmo de tipo “bee colony”. La función objetivo consiste en reducir el makespan, además los autores dan solución al problema en un tiempo polinómico y confirman que en el mercado no existe software que pueda ser parametrizado para problemas tan puntuales.

El alcance de los problemas de secuenciación de máquinas no relacionadas abarca todo tipo de industrias, como la perforación de pozos petroleros, tal es el caso de Avdeenko et al. (2017). La cantidad de recursos primarios y secundarios tan alto

crea un problema de secuenciación complejo para los autores. El problema se solucionó usando modelamiento matemático y algoritmos computacionales, logrando soluciones iguales a los métodos exactos en más de la mitad de las instancias.

Cuando un problema es muy complejo, se puede optar por la modificación de los algoritmos poblacionales como hicieron Kılıç et al. (2018), en este proyecto se modificó un algoritmo poblacional de tipo “ant-lion ALO”, además los autores cambiaron el método de selección de tipo ruleta por el método de sorteo. Con esta modificación lograron minimizar el makespan, se destaca la adición de tiempo de Set Up de recursos en las restricciones.

En el trabajo de Chergui et al. (2018) se propone una función objetivo para minimizar el incumplimiento de fechas de entrega, en este caso fue necesario crear una heurística totalmente nueva para dar solución al problema.

Un proyecto similar al anterior soluciona un problema de secuenciación en una planta de inyección, este proyecto realizado por Şafak et al. (2019) consideró restricciones de mano de obra y tiempos de Set Up de recursos primarios y secundarios, realmente un enfoque innovador. Los autores desarrollaron una heurística de descomposición, obteniendo soluciones de calidad que no fueron posibles con el método exacto.

Con el desarrollo de nuevas restricciones por la diversidad de investigaciones, se han desarrollado modelos como el propuesto por Dang (2020), que también tiene en cuenta los tiempos de Set Up de recursos comunes y su reemplazo. El objetivo de este modelo fue reducir la tardanza total y los tiempos de alistamiento de los recursos, planteamiento necesario por su gran volumen. El autor combinó un modelo de programación entera con un algoritmo genético, creando un algoritmo robusto. La matheuristica propuesta es 30% mejor encontrando soluciones para todas las instancias que el modelo usado por la compañía basado en programación entera.

Las organizaciones también buscan mejorar el desempeño de su mano de obra, un enfoque innovador fue el trabajo de Liu et al. (2019) que buscó maximizar el aporte de los trabajadores especializados, Este problema maximizó el cumplimiento de las tareas asignadas a los trabajadores y el cumplimiento de los trabajos. Es innovador en la medida que usa los trabajadores para los alistamientos de las máquinas.

Autores como Sarac et al. (2017) mencionan que existen problemas que pueden ser resueltos sin llegar a soluciones de calidad. En este caso los autores formularon un

algoritmo basado en técnicas lexicográficas para programar los trabajos en una planta de inyección, innovaron en añadir restricciones de elegibilidad para que los moldes no se superpongan en las máquinas.

Desarrollar directamente heurísticas para problemas de máquinas no relacionadas es otra opción, Yepes et al. (2020) desarrollaron directamente su trabajo usando heurísticas. En este caso, al igual que el anterior, los autores consideran tiempos de Set Up y recursos no renovables. La función objetivo tiene dos elementos, minimizar el makespan y el número de recursos necesarios para la operación, este proyecto es innovador porque considera el costo de operación.

La producción en algunas empresas requiere que se interrumpan los trabajos para realizar tareas adicionales o ajustes en los recursos (Jiang et al, 2020). Es este caso los autores desarrollaron programas de secuenciación ideales porque minimizan el makespan y tiempo total de terminación de los trabajos, este proyecto es innovador porque además de las reglas de retención se buscó maximizar la satisfacción del cliente. Luego de realizar una evaluación de la literatura actual, es evidente la necesidad de aplicar una metaheurística para solucionar problemas de gran escala, sobre todo en el momento de secuenciar los trabajos en las máquinas.

En investigaciones recientes como la de Muter (2020) es común la utilización de varias técnicas de modelamiento, el autor utilizó en su trabajo modelos exactos, heurísticos y lo más innovador en este caso y a diferencia de toda la literatura es el uso de machine learning como método de solución.

Finalmente es necesario resaltar que no existen muchos trabajos aplicados a problemas de secuenciación de máquinas no relacionadas paralelas. No existen muchos proyectos de investigación relacionados con problemas de secuenciación para plantas de inyección, además pocos proyectos formulan funciones multiobjetivo. El presente trabajo de grado aborda precisamente esos tópicos omitidos por gran parte de la literatura existente justificando su desarrollo. Este proyecto de grado es interesante porque abordó el problema de secuenciación con instancias académicas y reales con información de producción.

3. DEFINICION DEL PROBLEMA

3.1. MODELO MATEMÁTICO

En este proyecto se formula un modelo matemático de programación lineal entera mixta (MILP) para representar el comportamiento del sistema de producción actual con el fin de reducir el makespan y tardanza.

Usando la nomenclatura convencional de secuenciación presentada en la Sección 2.1.1., el modelo matemático que representa el sistema de producción es $\boldsymbol{R_m \mid S_{jk}, M_j \mid Cmax, T_j}$. Esta nomenclatura índica que el problema es de secuenciación de máquinas no relacionadas, con restricciones de tiempos de alistamiento y restricciones de elegibilidad en máquinas y recursos.

El modelo MILP propuesto incluye dos objetivos tradicionales: minimizar el $Cmax$ y la tardanza respectivamente. Estos dos objetivos se representan en una suma ponderada en la función objetivo del modelo. Se seleccionaron como objetivos el $Cmax$ y la tardanza porque son los indicadores más importantes para la dirección de la planta de inyección, basados en estos indicadores se toman decisiones de piso y estratégicas para cumplir el requerimiento de planeación de la demanda y ventas.

Además, se tienen los siguientes supuestos: el procesamiento de los trabajos no permite que paren y sean reanudados en otra máquina, los procesos no comparten moldes para terminar un trabajo, se requiere un tiempo de alistamiento para el cambio de máquina, molde y material cuando se cambia un trabajo en específico. En la Tabla 1 se presenta la notación usada para formular el modelo MILP.

Tabla 1. Definición de parámetros y variables para el modelo MILP

Conjuntos	
N	Conjunto de trabajos
N_0	Conjunto de trabajos que incluye trabajo dummy que define inicio y fin de secuenciación.
M	Conjunto de máquinas.
H	Conjunto de moldes.
H_0	Conjunto de moldes, incluye moldes dummy usados para iniciar y terminar secuencias.
Subíndices	
j, k, l	Trabajos para secuenciar. $j, k, l = 0, 1, \dots, n$
i	Máquinas de inyección. $i = 1, \dots, q$
h, r	Moldes de inyección. $h = 1, \dots, H$

Tabla 1. (Continuación)

Variables	
C_{max}	Tiempo de terminación máximo.
T_j	Tardanza del trabajo j.
x_{ihjk}	Variable binaria para establecer la secuencia del trabajo, en este caso el trabajo j es anterior al trabajo k y se realiza en la máquina i con el molde h.
y_{ihk}	Variable binaria para definir la asignación del trabajo *k* a la máquina i con el molde h.
C_i	Tiempo de terminación del trabajo i.
FP_{jkh}	Es una variable auxiliar binaria que toma valor de uno cuando el tiempo de terminación del trabajo k es mayor que el tiempo de terminación del trabajo j.
Parámetros	
α	Peso del componente Cmax en el valor total de la función objetivo embebida.
β	Peso del componente Tj en el valor total de la función objetivo embebida.
p_{hk},	Tiempo del procesamiento del trabajo k con el molde h. Algunos moldes tienen réplicas más eficientes.
φ	$Big\ M.$
d_j	Tiempo de entrega del trabajo j.
S_{jk}	Tiempo de alistamiento para recursos por cambio de trabajo j a k.
$Elig(i,h,j)$	Matriz de compatibilidad entre los trabajos j, molde h y máquina i. No todos los moldes son compatibles con todas las máquinas.

Fuente: Autor.

El modelo matemático de tipo MILP es el siguiente:

$$Minimizar\ f(x) = \alpha * C_{max} + \beta * \sum_{j=1}^{n} T_j \quad \textbf{(1)}$$

Sujeto a:

$$\sum_{h=1}^{H}\sum_{j=1}^{n}\sum_{k=1}^{n} s_{jk} * x_{ihjk} + \sum_{h=1}^{H}\sum_{k=1}^{n} p_{hk} * y_{ihk} \leq C_{max},\ \ \forall i \in M, j \neq k \quad \textbf{(2)}$$

$$\sum_{h=0}^{H}\ \sum_{k=1}^{n} x_{ih0k} \leq 1, \qquad \forall i \in M \quad \textbf{(3)}$$

$$\sum_{i=1}^{q}\sum_{h=1}^{H}\sum_{k=0}^{n} x_{ih0k} = 0 \quad \textbf{(4)}$$

$$\sum_{h=1}^{H}\sum_{i=1}^{q} y_{ihk} = 1 \quad , \qquad \forall k \in N, \tag{5}$$

$$y_{ihk} = \sum_{j=0}^{n} x_{ihkj}, \qquad \forall i \in M, \forall h \in H, \forall k \in N, j \neq k \tag{6}$$

$$\sum_{h=1}^{H} y_{ihk} = \sum_{h=1}^{H}\sum_{j=0}^{n} x_{ihjk}\,, \qquad \forall i \in M, \forall k \in N, j \neq k \tag{7}$$

$$C_k - C_j + \varphi * \left(1 - x_{ihjk}\right) \geq S_{jk} * x_{ihjk} + \sum_{r=1}^{H} y_{irk} * p_{rk}\ , \qquad \forall j \in\ N_0\ , \forall k \in N, \forall h \in H, \forall i \in M, j \neq k \tag{8}$$

$$C_k\ \leq C_{max}, \qquad \forall k \in N \tag{9}$$

$$\sum_{k}^{n} x_{ihjk}\ \leq Elig_{ihj}, \qquad \forall i \in M, \forall j \in N, \forall h \in H \tag{10}$$

$$T_j \geq C_k - d_k, \qquad \forall j, k \in N \tag{11}$$

$$C_k - C_j + \varphi * \left(1 - FP_{jkh}\right) \geq \sum_{i=1}^{q}(y_{ihk} * p_{hk}) + \sum_{i=1}^{q}\sum_{r=1}^{H}\sum_{l=0}^{n} x_{irlk} * S_{lk}, \quad \forall j \in N_0, k \in N, \forall h \in H \tag{12}$$

$$FP_{jkh} + FP_{kjh} + \left(2 - \sum_{i=1}^{q} Y_{ihk} - \sum_{i=1}^{q} Y_{ihj}\right) \geq 1\ \ \forall j, k \in N\ , \forall h \in H \tag{13}$$

$$x_{ihjk}, y_{ihk}, FP_{kjh} \in \{0,1\}\ \ \forall i, j, h, k \tag{14}$$

$$C_j, T_j, C_{\max}\ \geq 0\ \forall\ j \tag{15}$$

Explicación de las ecuaciones y restricciones:

La ecuación (1**)** representa la función objetivo, los valores de α y β son ajustados según los intereses de gerencia y los valores de $Cmax$ y $tardanza$. Los valores de

α y β son: 0, 1, 0.7 y 0.3, se seleccionaron estos valores porque dan prioridad a uno u otro objetivo y se pueden usar para normalizar los valores de magnitud.

El conjunto de restricciones (2) definen el valor de $Cmax$ sujeto a los tiempos de proceso y de alistamiento entre los trabajos k y j en la máquina i.

El conjunto de restricciones (3) indica que existe un trabajo dummy que fija el inicio de la secuenciación de los trabajos en cada máquina i, permitiendo la identificación de los tiempos de alistamiento para la primera tarea programada.

El conjunto de restricciones (4) indica que la tarea dummy solo se puede realizar con el molde dummy.

El conjunto de restricciones (5) es usado para garantizar que un trabajo es asignado a una sola máquina y un solo molde.

El conjunto de restricciones (6) y (7) garantizan que un trabajo solo tiene un trabajo antecesor y un trabajo sucesor en cada máquina, incluyendo los trabajos y moldes dummy.

El conjunto de restricciones (8) está definido para calcular el tiempo de terminación del trabajo, garantizando que no se generen bucles de tareas.

El conjunto de restricciones (9) permite calcular el $cmax$, considera que debe ser mayor que el valor del tiempo de terminación de los trabajos j.

El conjunto de restricciones (10) es usada para garantizar que, por razones técnicas, los trabajos solo pueden ser procesados en ciertas máquinas y moldes. La matriz de compatibilidad tiene valores de 1 cuando existe compatibilidad y 0 cuando no existe compatibilidad entre trabajos y recursos.

El conjunto de restricciones (11) es usada para calcular la tardanza del trabajo j.

El conjunto de restricciones (12) indica que el tiempo de procesamiento del trabajo k es menor que la diferencia entre los tiempos de terminación de los trabajos k y j, cuando el tiempo de terminación del trabajo k es mayor que el del trabajo j. Este conjunto de restricciones también es útil para ajustar los tiempos de terminación en caso de que se deba esperar por un molde que está en uso en otra máquina, evitando el overlapping.

El conjunto de restricciones (*13*) guía la variable auxiliar FP, indicando si un trabajo k se realiza luego del trabajo j con el mismo molde h, independientemente de si están asignados en la misma máquina o si son tareas consecutivas.

Las restricciones (14) y (15) indican la naturaleza de las variables.

4. METODO DE SOLUCIÓN

Como se mencionó en la Sección 2, los algoritmos genéticos son uno de los métodos usados para solucionar problemas de secuenciación en máquinas no relacionadas paralelas o cualquier problema de secuenciación. Este método se caracteriza por encontrar buenas soluciones en tiempos razonables. Por tal motivo se escogió en este proyecto como una alternativa para solucionar el problema de secuenciación.

4.1. ESTRUCTURA DE ALGORITMO GENÉTICO PROPUESTO

El algoritmo genético está basado en el modelo canónico propuesto por Holland et al. (1975). Este modelo, consiste en replicar las propiedades evolutivas de los organismos para encontrar soluciones a problemas complejos.

El algoritmo genético propuesto específicamente para este proyecto está compuesto por:

- Cromosomas: contiene una solución potencial. La descripción de cromosoma diseñado se presenta en la Sección 4.2.
- Población: conjunto de cromosomas, su explicación está en la Sección 4.3.
- Función fitness: evalúa el valor de la función objetivo para cada cromosoma. El cálculo del fitness para este problema se explica en Sección 4.4.
- Métodos de crossover y mutación: usados para generar nuevas y mejores soluciones en cada generación. Los métodos usados son crossover de un punto, crossover uniforme y mutación de varios genes. Los crossovers están explicados en la Sección 4.5 y el método de mutación está en la Sección 4.6.
- Método de selección: aleatorio sin remplazo, método explicado en la Sección 4.7.

En Algoritmo 1, se presenta el pseudocódigo del algoritmo genético propuesto.

Algoritmo 1. Algoritmo genético propuesto.

$Procedure$: $Genetic\ Algorithm$
$Input$: P // *Población*
rnd_number // *Número aleatorio*
$mutation\_probability$: // *Problabilidad de mutación*
$NumGenerations$ // *Numero de generaciones*
$NumGen\_No\_Improv$ // *Numero de generaciones sin mejora*
$Output$: $Best_{score},\ jobs\ sequence, scheduling\ problem\ solution$

1. $t_{0 =} 0$
2. $P = \emptyset$
3. $Best\_score = \infty$
4. $P \leftarrow Initiate\_first\_population(n)$ // *Iniciar población tamaño n*
5. **for** *j= 1 to* $NumGenerations$ **do:**
6. **for** *k=1 to n/2* **do:**
7. $par_1, par_2 \leftarrow SelectParents(P)$ //*Select randomly* 2 *parents from P*
8. $son_1, son_2 \leftarrow 1PointCrossover(par_1, par_1)$ //*Apply* 1 *point crossover*
9. **for** $i = 1$ *to* 2 **do**:
10. **if** $rnd\_number < mut\_probability$:
12. $son_i = mutation(son_i)$
13. **End if**
14. **Next i**
15. $P = P \cup \{son_1, son_2\}$
16. **Next k**
17. **for** *k=1 to n/2* **do:**
18. $par_1, par_2 \leftarrow SelectParents(P)$ //*Select randomly* 2 *parents from P*
19. $son_1, son_2 \leftarrow UniformCrossover(par_1, par_1)$ //*Apply uniform crossover*
20. **for** *i=1 to 2 do:*
21. **if** $rnd\_number < mut\_probability$:
22. $son_i = mutation(son_i)$
23. **End if**
24. **Next i**
25. $P = P \cup \{son_1, son_2\}$
26. **Next k**
27. $P \leftarrow SortByFitness(P)$
28. $P \leftarrow SelectNextGeneration(P)$
29. **if** $best\_fitness(P) = Best\_score$
30. $repeat = repeat + 1$
31. **Else**
32. $repeat = 0$
33. $Best\_score \leftarrow best\_fitness(P)$
34. **End if**
35. **if** $repeat = NumGen\_No\_Improv$
36. *j=* $NumGenerations$
37. **End if**
38. **Next j**
39. **return** $Best\_score$

End $Procedure$

Fuente: Autor.

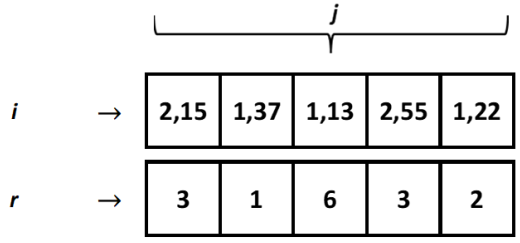
El AG (algoritmo genético), genera inicialmente una población P de n individuos (línea 4), los individuos corresponden a cromosomas creados siguiendo un método de random-keys que se explica en la Sección 4.3. Luego se empiezan a crear las diferentes generaciones a partir de los mecanismos de crossover y mutación. Hay dos tipos de crossovers: de un punto y uniforme (Ver Sección 4.5), aleatoriamente se eligen dos padres sin reemplazo (líneas 7 y 18), de cada crossover se generan $\frac{n}{2}$ nuevos individuos el crossover de un punto se observa en la línea 8 y la aplicación del crossover uniforme se observa en la línea 19. A cada nuevo individuo se le somete al mecanismo de mutación (Ver Sección 4.6) con una probabilidad $mut\_probability$ (líneas 12 y 22).

Cada nuevo individuo es incluido en la población (líneas 15 y 25). Luego, se hace la selección de los mejores individuos (Ver Sección 4.7) según el menor valor de fitness, esta selección representa la nueva generación (línea 33). El algoritmo va guardando el mejor valor de cada generación hasta que una de las siguientes dos condiciones se cumpla: un número $NumGen\_No\_Improv$ de generaciones sin que se obtenga una nueva mejor solución, o hasta que se hayan generado un número $numGenenerations$ de generaciones.

4.2. CROMOSOMAS

El cromosoma se diseñó para representar los trabajos y su requerimiento de máquinas y moldes sujeto a todas las restricciones propuestas. El diseño evita la posibilidad de presentar incompatibilidades durante mutaciones o crossovers. El diseño es de dos filas como se observa en la Figura 1.

Figura 1. Ejemplo de un cromosoma



Fuente: autor.

Cada columna representa un gen ligado a un trabajo. Cada trabajo es asignado a un molde y a una máquina, la fila i representa las máquinas, cada valor de máquina está formado por dos casillas, una compuesta por un número real donde la parte entera representa la máquina y la parte decimal es usada para, en orden ascendente, generar la secuencia de cada trabajo. La fila r de la Figura 1

representa el molde asignado para cada trabajo. De forma ilustrativa se presenta en la Figura 2, el cromosoma ordenado de forma ascendente según la parte decimal del valor máquina asignado. El diseño del cromosoma y el algoritmo propuesto permiten experimentar con el método sin que se presente overlappings durante la generación de cromosomas, poblaciones o en la aplicación de métodos de mutación y crossover.

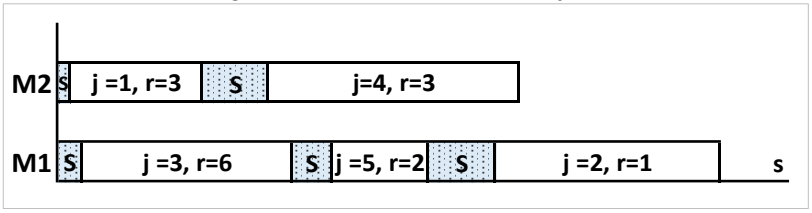
Figura 2. Cromosoma ordenado.

Secuencia →	1	2	3	1	2
j →	3	5	2	1	4
i →	1,13	1,22	1,37	2,15	2,55
r →	6	2	1	3	3

Fuente: autor.

Los trabajos 3, 5 y 2 se secuencian en la máquina 1 en la secuencia 1-2-3, con los moldes 6, 2 y 1 respectivamente. Los trabajos 1 y 4 se secuencian en la máquina 2 en el orden 1-2 usando los moldes 3 en cada trabajo. En la Figura 3 se presenta la secuencia de los trabajos i en cada máquina con su respectivo molde r, las barras de colores con letra “S” representan los tiempos de alistamiento.

Figura 3. Secuenciación trabajos.



Fuente: autor.

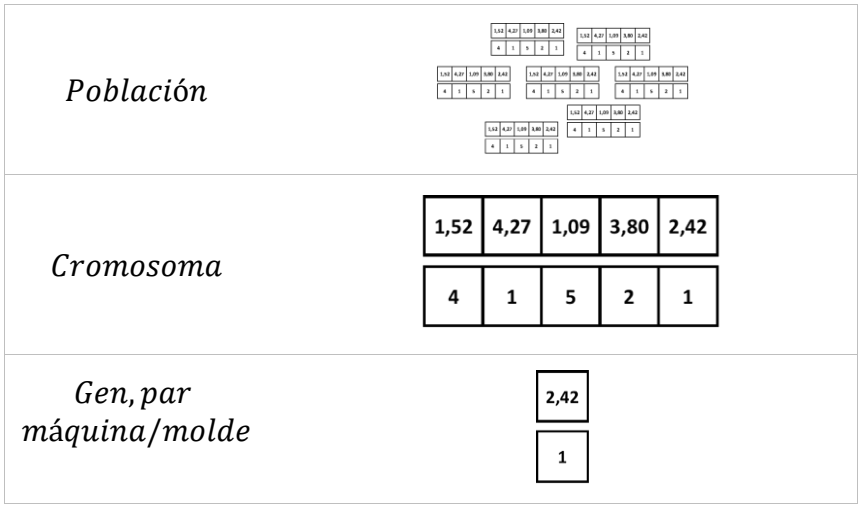
4.3. GENERACIÓN DE POBLACIÓN INICIAL

La población inicial es un conjunto de n cromosomas creados aleatoriamente y que contienen la información referente a máquinas, moldes y tiempos, sujetos siempre a las restricciones de elegibilidad. El método random-keys es usado para asignar las máquinas, se crea un numero racional, la parte entera de este número corresponde a la máquina y la parte decimal es usada para secuenciar el trabajo

como se mencionó en la Sección 4.2. Los moldes son asignados aleatoriamente y también están sujetos a restricciones de elegibilidad.

En la Figura 4, se puede observar los componentes de una población.

Figura 4. Componentes de una población.



Fuente: Autor.

La población se conserva como un conjunto y se le aplica cada uno de los mecanismos de crossover y mutación (este último de acuerdo con una probabilidad) del algoritmo genético propuesto para este proyecto.

4.4. DEFINICIÓN DE FUNCIÓN FITNESS

La función fitness calcula el valor de $Zmin$ para cada cromosoma, en este proyecto corresponde a:

$$Z_{min} = \alpha * Cmax + \beta * T(j)$$

Se desarrolló un procedimiento para calcular el fitness de cada cromosoma, asegurando que se cumplan las restricciones de elegibilidad y que no exista overlapping en la asignación de máquinas y moldes. El procedimiento se presenta en el Algoritmo 2, los elementos usados son iguales a los descritos en la Sección 3.1, elementos auxiliares tienen su descripción en forma comentario en el Algoritmo 2.

Algoritmo 2. Procedimiento para calcular fitness Zmin

$Procedure: Calcular\_Fitness(Chrom)$

$Input:$	$Chrom$	$// Cromosoma$
	α, β	
	p_{hk}	
	$S_{j,k}$	$// Tiempo\ alistamiento$
	d_j	
$Output:$	$Fitness(Chrom)$	$// Valor\ fitness\ del\ cromosoma.$
ϑ_j	$\leftarrow \emptyset$	$// Tiempo\ terminación\ inicial\ trabajo\ j.$
C_j	$\leftarrow \emptyset$	$// Tiempo\ de\ terminación\ trabajo\ j.$
Δ_C_j	$\leftarrow \emptyset$	$// Tiempo\ de\ terminación\ final\ trabajo\ j.$
γ_j	$\leftarrow \emptyset$	$// Tiempo\ de\ inicio\ trabajo\ j.$
T_j	$\leftarrow \emptyset$	$// Tardanza\ trabajo\ j.$
$rkey$	$\leftarrow i + u[0,1]$	$// random\ keys\ máquina.$
$Adjmt$		$// Ajuste\ C_j$

1. $Perform\ ascending\ sort\ to\ Chrom\ by\ \boldsymbol{random\_key}\ value: Sort(Chrom)$
2. $\vartheta_j = p_{h,k} + S_{j,k}$
3. $\gamma_{j=0} = 0\ \forall\ i \neq i - 1$
4. $\gamma_j = \vartheta_{j-1}\ \forall\ i = i - 1$
5. $C_j = \vartheta_j + \gamma_j$
6. ***While*** $Adjmt = true$ ***do***:
7. ***for*** $j = 1\ to\ n$ ***do***:
8. $Group\ j\ by\ r\ value\ and\ perform\ ascending\ sort\ to\ j\ by\ random\_key\ decimal\ value: Group(Sort(j))$
9. $\Delta\_C_j = C_j\ \forall\ r = 0, r \neq r - 1$
10. ***if*** $\gamma_j < \Delta\_C_{j-1}$:
11. $Adjmt = True$
12. $\Delta\_C_j = \max(\Delta\_C_{j-1} + \vartheta_j, \Delta\_C_j\ \forall\ j\ /\ i = i - 1)$
13. $Update\ \gamma_j: \gamma_j = \max(\Delta\_C_{j-1}, \vartheta_j)$
14. $Update\ C_j: C_j = \vartheta_j + \gamma_j$
15. ***else:***
16. $\Delta\_C_j = C_j$
17. ***end if***
18. ***next j***
19. ***end While***
20.
21. $Cmax = max\ (C_j)$
22. ***for*** $j = 1\ to\ n$ ***do***:
23. $T_j = \Sigma \max(0, C_j - d_j)$
24. ***next j***
25. $Fitness(Chrom) = \alpha * Cmax + \beta * T_j$
26. ***return*** $Fitness(Chrom)$

End $Procedure$

Fuente: Autor.

A modo de ejemplo se presenta el cálculo numérico del fitness para un cromosoma con 5 trabajos, 2 máquinas, 2 moldes (los datos para este ejemplo no corresponden a la solución de las instancias con optimalidad, son ilustrativos). El cálculo se realiza con el Algoritmo 2 (Ver Sección 4.4), el cromosoma diseñado para este ejemplo numérico se presenta en la Figura 5.

Figura 5. Cromosoma original.

1,5	1,4	1,7	1,2	2,1
1	2	1	2	1

Fuente: Autor.

1. Ordenar cromosoma(chrom).

Se ordena el cromosoma respecto al valor de la máquina según la línea 1 del Algoritmo 2. Se ordena de forma ascendente (ver Figura 6) según el valor real asignado para la máquina i.

Figura 6. Cromosoma ordenado.

1,2	1,4	1,5	1,7	2,1
2	2	1	1	1

Fuente: Autor.

2. Calcular: $\vartheta_j, \gamma_j, C_j$

Se calcula cada elemento según las líneas 2, 3, 4 y 5 del Algoritmo 2. Los valores calculados están en la Tabla 2. Cuando un trabajo inicia la operación de un molde el valor de inicio es cero.

Tabla 2. Cálculo de $\vartheta_j, \gamma_j, C_j$

j	ϑ_j	γ_j	C_j
4	32	0	32
2	54	32	86
1	49	86	135
3	25	135	160
5	150	0	150

Fuente: Autor.

3. Agrupar trabajos por valor de molde.

Los trabajos son agrupados (Ver Tabla 3) según la línea 8 del Algoritmo 2, acorde al valor numérico entero asignado como molde, en esta instancia existen dos grupos, uno para el molde uno y otro para el molde dos.

Tabla 3. Agrupar trabajos por valor de molde.

j	r	i	ϑ_j	γ_j	C_j
1	1	1,5	49	86	135
3	1	1,7	25	135	160
5	1	2,1	150	0	150
4	2	1,2	32	0	32
2	2	1,4	54	32	86

Fuente: Autor.

4. Ordenar trabajos respecto a valor decimal de máquina.

El proceso (Ver Tabla 4) se realiza según la línea 8 del Algoritmo 2. Primero se toma el número real asignado a cada trabajo como valor máquina y se retira la parte entera, luego con la parte decimal se ordena de menor a mayor cada trabajo, respetando el agrupamiento por moldes previo.

Tabla 4. Ordenar trabajo respecto a valor decimal de máquina.

j	r	i	ϑ_j	γ_j	C_j
5	1	0,1	150	0	150
1	1	0,5	49	86	135
3	1	0,7	25	135	160
4	2	0,2	32	0	32
2	2	0,4	54	32	86

Fuente: Autor.

5. Evaluar si ajuste de tiempo de terminación Δ_C_j es necesario.

La evaluación (Ver Tabla 5) se hace con las líneas 10 y 11 del Algoritmo 2. En este caso solo los valores de $C(1) y\, C(3)$ tienen que ser ajustados. El nuevo valor de Δ_C_j se ajusta con la línea 12 del Algoritmo 2.

Tabla 5. Evaluar ajuste de C(j)

j	r	i	ϑ_j	γ_j	C_j	$Ajustar?$	Δ_C_j
5	1	0,1	150	0	150	No	150
1	1	0,5	49	86	135	Si	199
3	1	0,7	25	135	160	Si	224
4	2	0,2	32	0	32	No	32
2	2	0,4	54	32	86	No	86

Fuente: Autor.

Si no es necesario ajustar el valor de Δ_C_j entonces Δ_C_j toma el valor descrito en la línea 15 del Algoritmo 2. Luego se realiza el ajuste del valor de Inicio del trabajo j (γ_j) usando la línea 13 del Algoritmo 2. También se actualiza el valor de $C(j)$ usando la línea 14 del Algoritmo 2.

Tabla 6. Ajustar valor de γ_j & $C(j)$.

j	ϑ_j	γ_j	C_j
4	32	0	32
2	54	32	86
1	49	150	199
3	25	199	224
5	150	0	150

Fuente: Autor.

Luego el bucle que inicia en la línea 6 del Algoritmo 2 se repite, si el valor de ajuste es no, entonces termina el bucle porque no es necesario hacer más ajustes.

Tabla 7. Repetir proceso de cálculo para $\vartheta_j, \gamma_j, C_j$

j	r	i	ϑ_j	γ_j	C_j	$Ajustar$	Δ_C_j
5	1	0,1	150	0	150	No	150
1	1	0,5	49	150	199	No	199
3	1	0,7	25	199	224	No	224
4	2	0,2	32	0	32	No	32
2	2	0,4	54	32	86	No	86

Fuente: Autor.

6. Calcular $Cmax$.

Se calcula el máximo tiempo de terminación $Cmax$ usando la línea 21 del Algoritmo 2.

$$Cmax = max(\Delta\_C_j)) = 224$$

7. Calcular $Tardiness\ T(j)$.

Se calcula la tardanza con la línea 23 del Algoritmo 2.

$$d(j) = (50, 150, 85, 49, 87)$$
$$T(j) = \Delta\_C_j - dj$$
$$T(j) = (100, 49, 139, 0, 0)$$

8. Calcular valor función fitness.

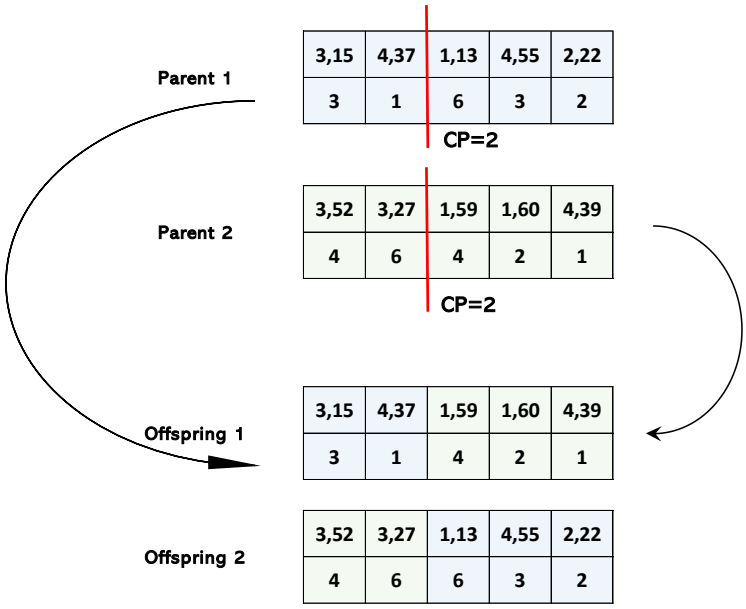
Por último, se calcula el valor del $fitness$ del cromosoma o $Zmin$ usando la línea 25 del Algoritmo 2. Para este ejemplo se usa un valor $\alpha = 0.7$ y $\beta = 0.3$

$$Zmin = 0.7 * Cmax + 0.3 * \Sigma T(j)$$
$$Zmin = 0.7 * 224 + 0.3 * 288$$
$$Zmin = 243{,}2\ segundos.$$

4.5. CROSSOVERS

Para la generación de nuevos individuos, en el algoritmo propuesto se aplicó a los miembros de la población mecanismos de crossover de un punto y crossover uniforme. En el crossover de un punto, primero se seleccionan dos padres (cromosomas) de la población, la selección es aleatoria sin reemplazo. Luego se crea un punto aleatorio para dividir los dos cromosomas en ese punto especifico y para combinarlos creando dos offsprings. En la Figura 7 se presenta la aplicación del crossover de un punto a dos cromosomas.

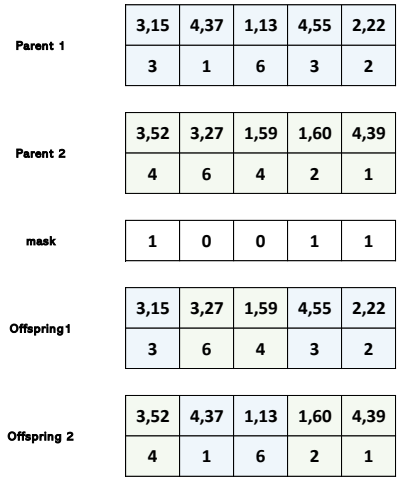
Figura 7. Aplicación de crossover de 1 punto.



Fuente: Autor.

En el crossover uniforme (Ver Figura 8), se crea un vector de valores 0 y 1 usando un generador de números pseudoaleatorios uniforme, este vector tiene igual longitud que el cromosoma, si el valor es 1 el valor del gen para el offspring 1 corresponde al padre 1, si el valor es 0 el valor del gen para offspring 1 corresponde al padre 2. El mecanismo es aplicado en forma contraria para el offspring 2.

Figura 8. Aplicación de crossover uniforme.

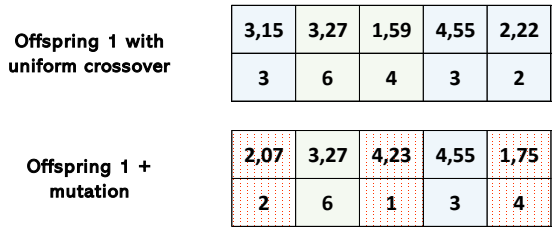


Fuente: Autor.

4.6. MUTACIÓN

El cromosoma puede mutar en 1, 2 o 3 de sus genes de forma aleatoria, sujeto al parámetro $mutation\_probability$. Un numero aleatorio uniforme es generado, si el valor del aleatorio es menor a la probabilidad de mutación se aplica la mutación a: moldes, máquinas y valor decimal del número real asignado como valor máquina, de lo contrario el cromosoma conserva la estructura definida luego de aplicar un mecanismo de crossover. El método garantiza que se cumplan las restricciones de elegibilidad. En la Figura 9, se presenta un ejemplo de la mutación usada.

Figura 9. Aplicación mecanismo de mutación a 3 genes.

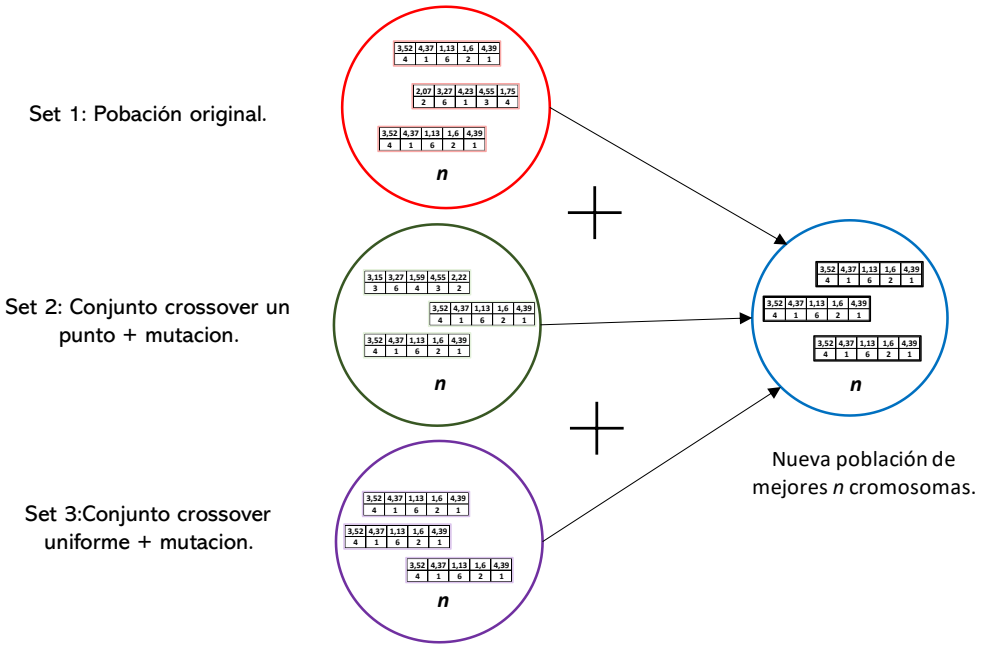


Fuente: Autor.

4.7. SELECCIÓN DE NUEVA POBLACIÓN

El algoritmo selecciona la nueva población a partir del conjunto formado por: población inicial, offsprings de crossover de un punto, offsprings de crossover uniforme y sus respectivas mutaciones. Por ejemplo, para una población inicial de 600 cromosomas, el offspring de un punto crea 600 cromosomas, el crossover uniforme crea 600 cromosomas también, la nueva población es de 1800 cromosomas (las mutaciones ya hacen parte de los conjuntos resultados de los crossovers). Al conjunto de 1800 cromosomas se aplica la función fitness (Ver Sección 4.4) y se selecciona la nueva población con los mejores 600 valores de fitness. En la Figura 10 se presenta un diagrama con el método de selección. En este proyecto se tiene un principio de elitismo para transmitir las mejores características (fitness) a nuevas generaciones. El procedimiento que ordena la población acorde a los valores de fitness de menor a mayor también se encarga de transmitir los mejores cromosomas entre generaciones, actuando como principio de elitismo.

Figura 10. Selección de nueva población.



Fuente: Autor.

5. EXPERIMENTOS COMPUTACIONALES

Un conjunto de 221 experimentos computacionales fueron desarrollados para probar el modelo MILP y para medir el desempeño del algoritmo genético. En la literatura no existen instancias relacionadas, razón por la cual se crearon instancias basadas en la operación real de la planta de inyección.

Se experimentó con el método MILP y con el algoritmo genético, además se ajustaron los parámetros del algoritmo genético con diseño factorial (Ver Anexo A). Las instancias son usadas para resolver el problema de secuenciación con el método exacto, posteriormente los resultados son usados para probar el desempeño del algoritmo genético.

Como se mencionó en la Sección 3.1, se formularon dos objetivos que se representan en una suma ponderada, esta configuración resulta muy interesante para ver cómo se desempeñan los métodos de solución con diferentes valores de α y β, en la Tabla 8 se presentan los diferentes valores de función objetivo.

Tabla 8. Configuraciones de funciones objetivo.

Función objetivo	Configuración
$Zmin_1$	$\alpha * Cmax + \beta * \Sigma T(j)$
$Zmin_2$	$Cmax$
$Zmin_3$	$\Sigma T(j)$
$Zmin_4$	$\beta * Cmax + \alpha * \Sigma T(j)$

Fuente: Autor.

216 experimentos son de instancias random y 4 experimentos con instancias basadas en información de producción real. Finalmente se solucionó un caso real, que es una instancia diseñada con datos de producción de la planta de inyección, este caso real representa la programación de la producción de piezas tipo A para un mes.

5.1. DESCRIPCIÓN ELABORACIÓN DE INSTANCIAS

Se plantearon instancias random para ser solucionadas con el modelo MILP y el algoritmo genético. Los parámetros fueron configurados de forma similar, que en el trabajo de Bektur et al. (2019). El trabajo de Kim et al (2007) fue el referente para definir la fórmula para calcular las fechas de entrega $d(j)$, se usó una distribución uniforme discreta, con parámetros de ajuste llamados $\omega\,(0{,}1)$ y $\sigma(1{,}2)$ que flexibilizan o ajustan la fecha de entrega. En la Tabla 9 se presenta un resumen de los parámetros usados en la generación de las instancias.

Tabla 9. Configuración de instancias random.

Ítem	Parámetro	Descripción	Distribución / Valor	Alternativas
1	i	Máquina	$2, 3, 5$	3
2	h	Moldes	$2, 4, 6$	3
3	j	Trabajos	$3 * j$	3
4	$p(h, j)$	Tiempo	$U(10, 100)$	1
5	$s(j)$	Set up	$U(5, 25)$	1
6	$elig(i, h, j)$	Matriz elegibilidad	$P(0.7)$	1
7	$d(j)$	Due date	$U[\omega * \sum(s(j,k) + p(j)/i), \sigma * \sum(s(j,k) + p(j)/i]$	1
8	Duplicidad molde	Prob. duplicado	$P(0.2)$	1
9	$p(h, j)$	Tiempo réplica	$P(h, j) * 0{,}9$	1

Fuente: Autor.

La nomenclatura de las instancias usada es: $w = trabajo, m = máquina, r = molde$. Se crearon 27 tipos de instancias random con los tamaños presentados en la Tabla 10. La cantidad total de instancias *random* solucionadas es de 216, 108 solucionadas con el método exacto y 108 solucionadas con el algoritmo genético propuesto. Adicionalmente se solucionan 4 instancias con información real ver Tabla 11 y un caso real para un total de 221 instancias.

Tabla 10. Tamaño de instancias random.

W	M	R
5	2	2
10	3	4
15	5	6

Fuente: Autor.

En las instancias que replican escenarios de producción los tiempos de procesamiento, alistamiento y entrega usados para la experimentación corresponden a los tiempos de producción, no han sido alterados. Las restricciones de elegibilidad de recursos son las que se presentan en la planta de inyección. Los tamaños de las instancias reales se pueden consultar en la Tabla 11.

Tabla 11. Tamaño de instancias reales con datos de producción.

Trabajos	Máquina	Moldes
12	10	12
24	20	17
15	15	12
13	15	12

Fuente: Autor.

Para las instancias reales con escenarios de producción, los parámetros de producción usados se descargaron del ERP de la compañía, los datos se tabularon en Excel y fueron convertidos en un archivo ".gdx" para permitir la interfaz entre los libros de Excel, GAMS studio© y neosserver.

5.2. AJUSTE DE PARAMETROS DE ALGORITMO GENÉTICO

El éxito de la experimentación en un algoritmo genético depende en gran parte del ajuste correcto de sus parámetros. Para el algoritmo propuesto en este proyecto se realizó un diseño factorial de tipo $5^1\ 3^2$ (un factor de cinco niveles y 2 factores de 3 niveles) con el objetivo de encontrar los valores apropiados para el tamaño de la población, valor de crossover y valor de probabilidad de mutación.

Dado el carácter aleatorio del algoritmo se hicieron 270 experimentos con tres instancias, en el Anexo A se presenta detalladamente el diseño factorial. En la Tabla 12, se presentan los valores de los parámetros seleccionados, esta definición de parámetros permite obtener valores de fitness equiparables o mejores que los valores obtenidos con el método exacto de solución.

Tabla 12. Valores de parámetros ajustados para algoritmo genético.

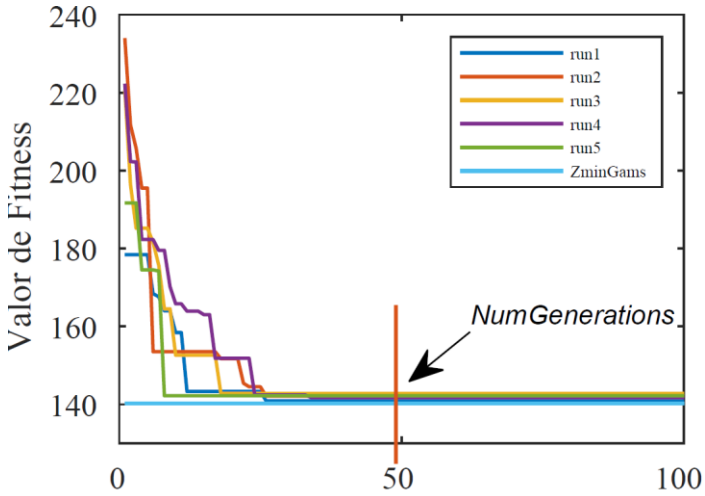
Parámetro	Valor
Population	600
Crossover	0.5
Mutation	0.15
Generaciones	50
Stop	15 repeticiones

Fuente: Autor.

Los parámetros número de generaciones y stop fueron definidos haciendo corridas con 400 generaciones, luego se graficaron los valores de fitness y se determinó visualmente que el algoritmo converge en un punto luego de 50 generaciones en promedio. En este punto de convergencia no se producen mayores cambios en el valor de fitness para futuras generaciones.

También se determinó que luego de obtener 15 mejores valores de fitness consecutivos no se produce un mejoramiento en el fitness. El proceso se repitió en instancias de tamaño pequeño, mediano y reales para validar su veracidad. La experimentación se realizó como se menciona en el Anexo A con 10 instancias, con este grupo de instancias se determinó el valor de parada. En la Figura 11 se puede observar el comportamiento estático del valor de fitness luego de un determinado número de generaciones, 50 generaciones de evolución es un parámetro universal que aplica a todo tamaño de instancias.

Figura 11. Solución instancia 10w5m6r con parámetros ajustados.



Fuente: Autor.

5.3. RESULTADOS DE MODELO MILP CON MÉTODO EXACTO

Las instancias random del modelo MILP fueron formuladas en Gams Studio 1.4.5 y solucionadas con la plataforma neosserver, utilizando el solver Cplex versión 20.1. Se estableció un gap de 0,000001%. El tiempo límite máximo de utilización de recursos (CPU time) es de 8 horas, límite fijado por defecto en la plataforma de neosserver. El servidor de neosserver asignado tiene un CPU con dos procesadores Intel Xeon E5-2698 de 2.3GHz (32 núcleos en total), memoria RAM de 192 GB.

La experimentación con el modelo matemático permitió validar que la secuenciación de los trabajos en las respectivas máquinas y moldes cumple con las restricciones. En la Tabla 13 se presentan los resultados del modelo MILP solucionado con el método exacto. El valor de la función objetivo $Zmin$, el valor de $Cmax$, la tardanza $T(j)$, el tiempo de solución τ, el tiempo de solución promedio de todas las instancias $\bar{\tau}$, todos están en segundos. También se presenta el valor en porcentaje GAP cuando se cumple el límite de 8 horas. Los valores de función objetivo $Zmin$ en negrilla corresponden a las soluciones con optimalidad.

Tabla 13. Solución de instancias random con MILP.

$Instanc.$	$Scenarios$																			
	$Zmin_1 = \alpha * Cmax + \beta * \Sigma T(j)$					$Zmin_2 = Cmax$					$Zmin_3 = \Sigma T(j)$					$Zmin_1 = \beta * Cmax + \alpha * \Sigma T(j)$				
	$Zmin_1$	Gap	$Cmax$	$T(j)$	τ	$Zmin_2$	Gap	$Cmax$	$T(j)$	τ	$Zmin_3$	Gap	$Cmax$	$T(j)$	τ	$Zmin_4$	Gap	$Cmax$	$T(j)$	τ
5w2m2r	**294,9**		273	346	0,036	**273**		273	346	0,022	**346**		273	346	0,042	**324,1**		273	346	0,034
5w2m4r	**196,3**		187	218	0,105	**186**		186	226	0,076	**217**		198	217	0,275	**208,7**		187	218	0,166
5w2m6r	**226,7**		212	261	0,136	**205**		205	278	0,044	**261**		212	261	0,901	**246,3**		212	261	0,256
5w3m2r	**158,4**		141	199	1,103	**140**		140	226	1,272	**194**		150	194	1,409	**180,8**		150	194	0,806
5w3m4r	**155,1**		120	237	0,061	**120**		120	262	0,041	**237**		120		0,067	**201,9**		120	237	0,081
5w3m6r	**163,4**		170	148	0,312	**165**		165	204	0,084	**148**		170	148	0,601	**154,6**		170	148	0,358
5w5m2r	**231,8**		197	313	0,067	**194**		194	333	0,038	**313**		197	313	0,067	**278,2**		197	313	0,064
5w5m4r	**131,7**		111	180	0,162	**111**		111	180	0,092	**180**		126	180	0,297	**159,3**		111	180	0,314
5w5m6r	**75,5**		89	44	0,276	**89**		89	52	0,14	**44**		97	44	0,419	**57,5**		89	44	0,506
10w2m2r	**709**		435	1351	74,676	**425**		425	1626	1,211	**1337**		455	1337	197,08	**1072**		455	1337	108,01
10w2m4r	**388**		307	580	2952,69	**306**		306	682	3,49	557	0,39			28800	**488**		327	557	8763,52
10w2m6r	**274**		292	232	6154,93	**283**		283	342	1,11	170	0,73			28800	222,8	0,43			28800
10w3m2r	**397,6**		280	672	50,859	**256**		256	864	2,16	**672**		280	672	55,78	**554,4**		280	672	77,55
10w3m4r	**247,3**		178	409	66,33	**171**		171	480	0,539	**409**		178	409	229,94	**339,7**		178	409	104,5
10w3m6r	**341,3**		233	594	6000,63	**220**		220	733	2,42	594	0,26			28800	**485,7**		233	594	7614,86
10w5m2r	**385,6**		292	604	44,06	**280**		280	950	11,98	**604**		292	604	97,38	**510**		292	604	54,86
10w5m4r	**232,1**		230	237	26,24	**193**		193	494	3,9	**237**		234	237	37,76	**234**		230	237	132,34
10w5m6r	**140,2**		154	108	261	**135**		135	188	18,77	**108**		237	108	7,93	**121,8**		154	108	219,94
15w2m2r		0,14			28797	**496**		496	1757	373,4		0,47			28800		0,37			28800
15w2m4r		0,33			28691	424	0,01			28178		1,00			-		0,49			-
15w2m6r		0,48			31598	**623**		623	244,15	6,07		0,81			28800		0,69			28412
15w3m2r		0,44			28800	372	0,29			28800		0,61			28800		0,43			28800
15w3m4r		0,06			28800	**437**		437	1447	0,312		0,53			28800		0,36			28800
15w3m6r		0,46			28800	**291**		291	932	614		1,00			28800		0,085			28800
15w5m2r		0,23			28800	448	0,07			28800		0,42			28800		0,37			28800
15w5m4r		0,5			4839	265	0,27			28800		0,59			28800		0,77			28800
15w5m6r		0,24			12809	**176**		176	309	25547,85		0,30			28800		0,36			28800
$\bar{\tau}$					8798,8					5228,4					12208					16611

Fuente: Autor.

Con la experimentación se observó que el modelo matemático MILP puede solucionar con optimalidad instancias de un tamaño máximo de 10 trabajos 5 máquinas y seis moldes, instancias más grandes no se logran solucionar en el tiempo límite fijado por neosserver (8 horas) y el gap es elevado, específicamente en instancias de 15 trabajos. En la Tabla 13 se presenta la solución de todas las instancias, el 68% de las instancias de la Tabla 13 se solucionaron de forma óptima, el 32% de las instancias que no se solucionaron a optimalidad tienen la característica de muchos moldes y máquinas disponibles para realizar el mismo trabajo.

Todas las instancias de cinco trabajos se solucionan en torno a un segundo, tienen bastante celeridad por la cantidad baja de combinaciones entre recursos y trabajos. Caso contrario son las instancias de diez trabajos, que tardan varias horas en ser solucionadas, esta tendencia se observa principalmente en las instancias con seis moldes. El tiempo promedio de solución para los diferentes escenarios ordenado de forma ascendente es: $Zmin2$, $Zmin1$, $Zmin4$ y $Zmin3$. es interesante que para el escenario 3 donde la función objetivo es la minimización de la tardanza los tiempos de solución son radicalmente mayores a los demás escenarios, debido a la jerarquía de complejidad de los ambientes máquina y los respectivos objetivos a minimizar.

Con los experimentos se observó que la mitad de los escenarios conservan similitud en los valores de $Cmax$ y $T(j)$, en la otra mitad se observaron cambios en los valores de $Cmax$ o $T(j)$, estos cambios se observan en los escenarios $Zmin_2, Zmin_3, Zmin_4$, porque el objetivo principal cambia frente al escenario $Zmin_1$.

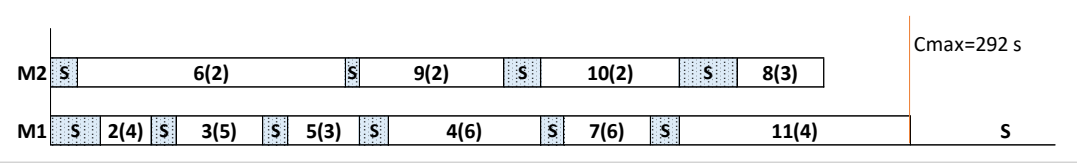
Profundizando en el párrafo anterior, la aproximación multiobjetivo del problema es necesaria porque con la experimentación se observó que en varias instancias los valores de $Cmax$ y $T(j)$ representan decisiones contradictorias en beneficio de una de las dos, por ejemplo, en la instancia $10w5m6r$ para el escenario $Zmin2$ los valores de $Cmax$ mejoran frente al escenario $Zmin1$ en detrimento de los valores de $T(j)$.

En términos generales se observa que las soluciones presentan valores razonables, no se aprecian resultados sesgados o ajenos a la información de entrada, se respetan en todas las instancias las matrices de elegibilidad entre los diferentes moldes, máquinas y trabajos, el autor realizó una comprobación manual de las soluciones validando que no se presenten overlappings.

5.4. REPRESENTACIÓN GRAFICA DE SOLUCIÓNES DE MÉTODO EXACTO

En la Figura 12, se presenta la secuenciación en el escenario $Zmin_1 = \alpha * Cmax + \beta * \Sigma T(j)$ de la instancia con 10 trabajos, 2 máquinas y seis moldes. Los cuadros de color azul con la letra S, corresponden a los tiempos de alistamiento por cambio de trabajos. En esta instancia el valor de la función objetivo es de 274 segundos, con un Cmax de 292 segundos y tardanza de 232 segundos, no existe gap.

Figura 12. Solución gráfica instancia de 10W2M6R.

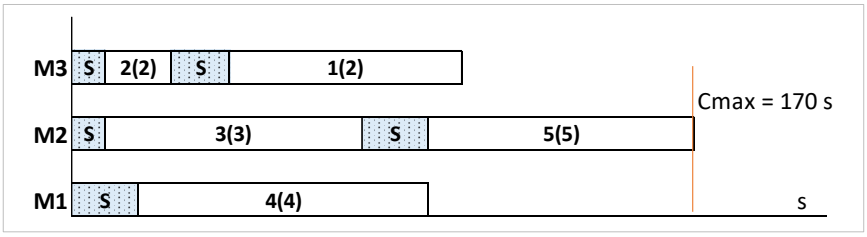


Fuente: Autor.

Se observa que no existe overlapping entre los diferentes recursos y trabajos, el Cmax está plenamente identificado y corresponde al trabajo 11. También se ven los tiempos de alistamiento, que cumplen con las definiciones de las instancias.

En la Figura 13, se presenta la gráfica de la instancia 5w3m6r, con función objetivo $Zmin_1 = \alpha * Cmax + \beta * \Sigma T(j)$. El valor de Cmax para esta instancia y función objetivo es de 170 segundos.

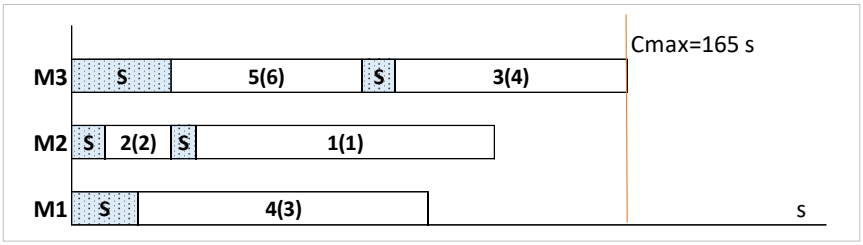
Figura 13. Solución gráfica 5w3m6r con Zmin1.



Fuente: Autor.

En la Figura 14, igualmente se presenta la solución de la instancia 5w3m6, pero esta vez se solucionó usando el escenario con función objetivo $Zmin_2 = Cmax$.

Figura 14. Solución gráfica 5w3m6r con Zmin2 (Cmax).



Fuente: Autor.

Lo que se puede observar y concluir de las soluciones graficas en la Figura 13 y Figura 14, es que el modelo MILP funciona correctamente, cuando se usa el escenario $Zmin_1$ el objetivo es minimizar la función embebida de $Cmax$ y $T(j)$ con sus respectivos pesos, en este caso el valor de $Cmax$ es de 170 segundos, pero cuando se soluciona la misma instancia con el escenario $Zmin_2$ que tiene como objetivo minimizar el $Cmax$, el valor de $Cmax$ baja a un valor de 165 segundos, respetando las restricciones de elegibilidad entre recursos y trabajos. Se observa que el modelo MILP explora las soluciones posibles y encuentra la mejor relación trabajo, molde, máquina y tiempos de proceso y alistamiento para encontrar la mejor solución del problema de secuenciación en instancias random de máximo 10 trabajos.

5.5. RESULTADOS DE INSTANCIAS REALES CON MÉTODO EXACTO

En la Tabla 14 se presenta la formulación de cuatro instancias con escenarios que se pueden presentar en la planta de inyección. $T(s)$ es el tiempo de solución en segundos.

Tabla 14. Resultados de instancias reales, con datos de producción.

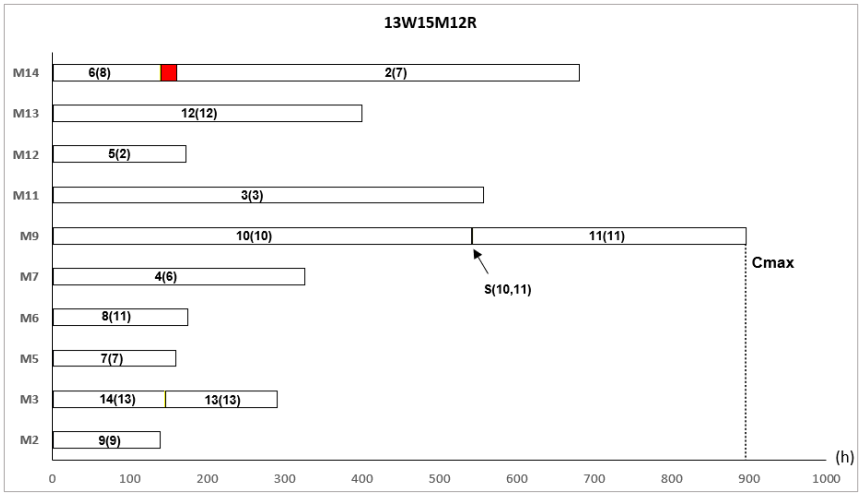
Instancias *wmr*			F.o.	Gap %	$T(s)$	Causa de terminación
Trabajos	Máquina	Moldes	Optimal			
12	10	12	**207,9**	0	0,661	
24	20	17		9,39	3020	Memoria limite
15	15	12		4,25	28800	Tiempo limite
13	15	12	**96,7**	0	96,7	

Fuente: Autor.

Solo se logran solucionar las instancias 1 y 4. La primera es un escenario muy sencillo porque existe una relación uno a uno entre trabajos, máquinas y moldes. La instancia número cuatro es un escenario intermedio con baja demanda, la instancia 2 y 3 son más cercanas a la realidad y no se logran solucionar con el

modelo MILP en tiempo razonable, debido a la complejidad del problema. En la Figura 15, se puede observar la solución de la instancia cuatro, el número en cada barra corresponde al trabajo y el número en paréntesis es el molde usado. No se presentan overlappings de recurso y solo aparece un idle time.

Figura 15. Solución instancia real, trece trabajos, quince máquinas, doce moldes.



Fuente: Autor.

En la máquina nueve se puede apreciar el Cmax de 896,7 horas que coincide con el tiempo de terminación del último trabajo y con mayor duración. Es muy importante anotar que el trabajo 11 solo se puede realizar en la máquina nueve con el molde once, este tipo de configuraciones molde/máquina se conoce como molde “esclavo” y son comunes en la planta de inyección. De esta forma se puede concluir que el modelo matemático MILP fue formulado correctamente, representando la dinámica de la planta de inyección. El modelo MILP puede solucionar instancias particulares y pequeñas del escenario de producción real. Por la complejidad del problema, la formulación MILP no puede solucionar las instancias reales más complejas y tampoco soluciona las instancias random con más de 10 trabajos, estos resultados justifican el desarrollo de un método heurístico para lograr buenas soluciones en tiempos razonables.

5.6. RESULTADOS DE INSTANCIAS USANDO ALGORITMO GENÉTICO

El conjunto de instancias de la Sección 5.1 se solucionó con el algoritmo genético propuesto para comparar el desempeño del método y medir la calidad de las respuestas. Las instancias fueron solucionadas usando una máquina virtual del

servicio Compute Engine de Google Cloud. La máquina virtual tiene 4 CPUs, con procesador Intel Xeon de 2.3 GHz, 16gb de memoria RAM y Windows server 2019. El algoritmo genético fue programado en Python 3.7.8, usando Eclipse como IDE. Cada instancia fue resuelta 10 veces. El mejor valor fitness de cada instancia solucionada con el algoritmo es representado por el símbolo $\boldsymbol{\delta_Z^{ga}}$**,** el tiempo está en segundos. El promedio de los resultados obtenidos en cada instancia solucionada (10 corridas) es representado por $\overline{\boldsymbol{\delta_Z^{ga}}}$ y el valor está en segundos. El tiempo promedio necesario para obtener cada solución es $\bar{\tau}$ y esta en segundos. Δ_Z^e denota el gap porcentual entre la mejor solución del algoritmo vs la solución del método exacto. Por último $\overline{\Delta_Z^e}$ representa el gap porcentual promedio de las soluciones del algoritmo versus las soluciones del método exacto. Los valores de Zmin, Cmax y tardiness están en segundos.

En la Tabla 15, se presentan los resultados con la función objetivo $Zmin_1 = \beta * Cmax + \alpha * \sum T(j)$.

Tabla 15. Resultados escenario $Zmin_1$ usando AG.

Instance	$\boldsymbol{\delta_Z^{ga}}$	$\overline{\boldsymbol{\delta_Z^{ga}}}$	$\bar{\tau}$	Δ_Z^e	$\overline{\Delta_Z^e}$
5w2m2r	**294,9**	**294,9**	5,28	0,00	0,00
5w2m4r	**196,3**	**196,48**	5,98	0,00	0,09
5w2m6r	**226,7**	226,96	4,71	0,00	0,11
5w3m2r	**158,4**	**158,4**	9,01	0,00	0,00
5w3m4r	**155,1**	**155,1**	9,008	0,00	0,00
5w3m6r	**163,4**	163,9	5,81	0,00	0,31
5w5m2r	**231,8**	**231,8**	9,54	0,00	0,00
5w5m4r	**131,7**	131,91	7,87	0,00	0,16
5w5m6r	**75,5**	**75,5**	6,44	0,00	0,00
10w2m2r	**709,8**	720,25	9,16	0,00	1,47
10w2m4r	**388,9**	401,39	6,076	0,00	3,21
10w2m6r	**274**	281,5	5,46	0,00	2,74
10w3m2r	**397,6**	410,9	13,1	0,00	3,35
10w3m4r	**247,3**	258,56	23,21	0,00	4,55
10w3m6r	**341,3**	357,7	9,29	0,00	4,81
10w5m2r	**385,6**	394,48	9,27	0,00	2,30
10w5m4r	234	247,99	13,93	0,82	6,85
10w5m6r	141,7	155,89	13,7	1,07	11,19
15w2m2r	**831,5**	866,92	31,7	**-0,05**	4,21
15w2m4r	**522,5**	572,4	27,81	**-0,04**	9,51
15w2m6r	**1323**	1415	18,88	**-2,72**	4,04
15w3m2r	**761,2**	798,11	13,91	**-3,85**	0,81
15w3m4r	647,7	675,8	35,05	0,67	5,04
15w3m6r	**418**	458,13	38,71	**-2,59**	6,77
15w5m2r	**707,3**	748,25	24,79	**-1,08**	4,65
15w5m4r	**230,5**	**276,85**	24,93	**-21,49**	**-5,71**
15w5m6r	**163,5**	199,15	24,86	**-10,75**	8,71
			avg.	**-1,48**	2,93

Fuente: Autor.

En los resultados de $Zmin_1$, se observa que el algoritmo genético propuesto iguala o mejora los resultados del método exacto. El 30% de las instancias tienen mejores resultados que el método exacto, específicamente son instancias que tenían gap en su solución con el método exacto. El tiempo de solución promedio para todas instancias usando el algoritmo genético propuesto es de 9,26 segundos, bastante menor comparado con 868,5 segundos del método exacto.

El gap promedio de las soluciones del algoritmo genético propuesto frente a las soluciones del método exacto es -1,48%. Se pueden lograr gap mínimos o nulos si se modifican los parámetros del GA, pero se necesitan mayores tiempos de solución. Podemos observar que existe un buen balance en encontrar buenas soluciones con un poco de Gap´s razonables. La parametrización de los datos de entrada del algoritmo genético es más versátil, si se compara con el método exacto. Es muy útil para instancias grandes.

Se puede apreciar un incremento en los tiempos de solución cuando las instancias tienen mayores combinaciones de recursos, aunque el incremento no es radical como ocurre en el método exacto. Los resultados más sobresalientes se observan en las instancias más grandes, el algoritmo genético supera al método exacto logrando mejores soluciones que el método exacto con tiempos de solución en torno a 26 segundos. Los resultados demuestran que el ajuste de los parámetros del Anexo A, fueron apropiados y universales para todas las instancias. En la Tabla 16, se presentan los resultados con la función objetivo $Zmin_2 = Cmax$

Tabla 16. Resultados escenario $Zmin_2$ usando AG.

Instance	δ_Z^{ga}	$\overline{\delta_Z^{ga}}$	$\bar{\tau}$	Δ_Z^e	$\overline{\Delta_Z^e}$
5w2m2r	**273**	**273**	9,54	0	0
5w2m4r	**187**	**187**	9,05	0	0
5w2m6r	**205**	**205**	7,46	0	0
5w3m2r	**140**	**140**	13,86	0	0
5w3m4r	**120**	**120**	12,54	0	0
5w3m6r	**165**	**165**	9,45	0	0
5w5m2r	**194**	**194**	14,08	0	0
5w5m4r	**111**	**111**	10,64	0	0
5w5m6r	**89**	**89**	10,28	0	0
10w2m2r	**425**	430,37	16,12	0	1,26
10w2m4r	307	311,74	12,87	0,33	1,88
10w2m6r	**283**	288,04	10,63	0,00	1,78
10w3m2r	**256**	257,28	32,42	0,00	0,50
10w3m4r	**171**	174,94	30,12	0,00	2,30
10w3m6r	**220**	224,22	45,24	0,00	1,92
10w5m2r	**280**	282,69	16,1	0,00	0,96
10w5m4r	**193**	203,65	26,49	0,00	5,52
10w5m6r	136	150,19	20,61	0,74	11,25
15w2m2r	502	510,98	25,37	1,17	2,98
15w2m4r	433	449,6	19,57	2,12	6,04
15w2m6r	**623**	659,02	20,39	0,00	5,78

Tabla 16. (Continuación)

Instance	δ_Z^{ga}	$\overline{\delta_Z^{ga}}$	$\bar{\tau}$	Δ_Z^e	$\overline{\Delta_Z^e}$
15w3m2r	**372**	392,35	17,66	0,00	5,47
15w3m4r	**437**	444,6	22,11	0,00	1,74
15w3m6r	302	318,26	20,36	3,78	9,37
15w5m2r	**448**	462,27	17,96	0,00	3,19
15w5m4r	263	283,37	17,76	**-0,75**	6,93
15w5m6r	187	209,17	21,27	6,25	18,85
			avg.	0,50	3,25

Fuente: Autor.

En el segundo escenario con $Zmin2$, el algoritmo genético logro igualar los resultados del método exacto en 78% de las instancias, es de resaltar que el Gap promedio de los mejores resultados del algoritmo genético frente al método exacto es de 0,5%. El algoritmo genético logro mejorar la solución del método exacto en la instancia $15w5m4r$, una de las pocas que no puedo solucionar con optimalidad el método exacto. En este escenario no se observan grandes mejoras porque son pocas las instancias con Gap en su solución del método exacto, igualmente en gran parte de las instancias el algoritmo genético iguala los resultados del método exacto.

El tiempo promedio de solución del conjunto de instancias fue de 18 segundos, bastante inferior al método exacto. Las instancias con diez trabajos o más son las que requieren más recursos CPU por la cantidad de combinaciones posibles. Se puede observar que los resultados del escenario $Zmin_2$ representan el modelo y están alineados con los resultados del método exacto.

En la Tabla 17, se presentan los resultados con la función objetivo $Zmin_3 = \sum T(j)$.

Tabla 17. Resultados escenario $Zmin_3$ usando AG.

Instance	δ_Z^{ga}	$\overline{\delta_Z^{ga}}$	$\bar{\tau}$	Δ_Z^e	$\overline{\Delta_Z^e}$
5w2m2r	**346**	**346**	7,54	0	0
5w2m4r	**217**	217,03	9,45	0	0,01
5w2m6r	**261**	**261**	4,51	0	0
5w3m2r	**194**	**194**	8,58	0	0
5w3m4r	**237**	**237**	7,79	0	0
5w3m6r	**148**	**148**	4,23	0	0
5w5m2r	**313**	**313**	8,81	0	0
5w5m4r	**180**	**180**	7,45	0	0
5w5m6r	**44**	**44**	6,24	0	0
10w2m2r	**1337**	1372	8,69	0	2,62
10w2m4r	**557**	586,12	9,46	0	5,23
10w2m6r	**170**	197,98	8,36	0	16,46
10w3m2r	**672**	724	13,08	0	7,74
10w3m4r	**409**	452,19	12,74	0	10,56
10w3m6r	**594**	636,43	10,58	0	7,14

Tabla 17. (Continuación)

Instance	δ_Z^{ga}	$\overline{\delta_Z^{ga}}$	$\bar{\tau}$	Δ_Z^e	$\overline{\Delta_Z^e}$
10w5m2r	**604**	625,71	12,97	0	3,59
10w5m4r	**237**	293,5	12,76	0	23,84
10w5m6r	**108**	122,03	12,37	0	12,99
15w2m2r	1521	1668,25	10,84	2,42	12,34
15w2m4r	**701**	851,24	11,99	**-0,99**	20,23
15w2m6r	**2837**	3090	12,09	**-5,40**	3,03
15w3m2r	**761,2**	798,11	13,91	**-1,19**	3,60
15w3m4r	**1038**	1150,76	14,39	**-0,38**	10,44
15w3m6r	**630**	766,99	12,7	**-8,83**	11,00
15w5m2r	**1287**	1378,75	14,38	**-1,30**	5,73
15w5m4r	**140**	245,57	14,55	**-13,58**	5,12
15w5m6r	**87**	146,79	14,28	**-15,53**	4,26
			Avg.	**-1,66**	6,15

Fuente: Autor.

En el escenario $Zmin3$, el algoritmo genético genera resultados iguales al método exacto en 96% de las instancias. El algoritmo genético mejora los resultados del método exacto en 30% de las instancias y los resultados más significativos están en las instancias de mayor tamaño. Los tiempos promedio de solución de las instancias usando el algoritmo genético son considerablemente menores a todos los escenarios, 10 segundos en promedio. Finalmente, en la Tabla 18, se presentan los resultados con la función objetivo $Zmin_4 = 0.3 * Cmax + 0.7 * \sum T(j)$.

Tabla 18. Resultados escenario $Zmin_4$ usando AG.

Instance	δ_Z^{ga}	$\overline{\delta_Z^{ga}}$	$\bar{\tau}$	Δ_Z^e	$\overline{\Delta_Z^e}$
5w2m2r	**324,1**	**324,1**	7,89	**0,00**	0,00
5w2m4r	**208,7**	**208,7**	10,16	**0,00**	0,00
5w2m6r	**246,3**	246,86	6,39	**0,00**	0,23
5w3m2r	**180,8**	**180,8**	13,09	**0,00**	0,00
5w3m4r	**201,9**	**201,9**	14,8	**0,00**	0,00
5w3m6r	**154,6**	**154,6**	8,09	**0,00**	0,00
5w5m2r	**278,2**	**278,2**	19,47	**0,00**	0,00
5w5m4r	**159,3**	159,72	25,65	**0,00**	0,26
5w5m6r	**57,5**	**57,5**	10,62	**0,00**	0,00
10w2m2r	**1072**	1090,53	19,08	**0,00**	1,73
10w2m4r	**488**	508,01	11,81	**0,00**	4,10
10w2m6r	**215**	233,19	10,19	**-3,50**	4,66
10w3m2r	**554,4**	576,08	20,56	**0,00**	3,91
10w3m4r	**339,7**	359,99	21,13	**0,00**	5,97
10w3m6r	**485,7**	505,6	46,6	**0,00**	4,10
10w5m2r	**510,2**	519,26	18,3	**0,00**	1,78
10w5m4r	**234,9**	273,9	18,36	**0,00**	16,60
10w5m6r	**121,8**	136,83	17,57	**0,00**	12,34
15w2m2r	1239,3	1318,36	25,92	2,09	8,61
15w2m4r	**615,7**	758,43	13,32	**-5,26**	16,70
15w2m6r	**2167**	2378,64	12,9	**-3,24**	6,21

Tabla 18. (Continuación)

Instance	δ_z^{ga}	$\overline{\delta_z^{ga}}$	$\bar{\tau}$	Δ_z^e	$\overline{\Delta_z^e}$
15w3m2r	**1229,2**	1305,32	19,72	**-3,49**	2,48
15w3m4r	896,4	983,49	14,29	0,86	10,65
15w3m6r	**589,6**	661,68	13,09	**-1,19**	10,89
15w5m2r	**1039,7**	1123,11	28,4	**-1,63**	6,26
15w5m4r	**185,8**	**278,77**	25,16	**-36,89**	-5,31
15w5m6r	**117,6**	174,4	32,05	**-21,18**	16,89
			Avg.	**-2,01**	4,78

Fuente: Autor.

En el escenario $Zmin_4$ el algoritmo genético iguala o mejora los resultados del método exacto en 93% de las instancias. Es de resaltar que 30% de las instancias solucionadas con el algoritmo genético propuesto tienen mejores soluciones que el método exacto, es de resaltar que la instancia $15w5m4r$ tiene el mayor mejoramiento a pesar de que es la que tiene el segundo mayor componente de combinaciones en recursos. Los tiempos de solución del algoritmo genético propuesto, crecen de forma razonable con el incremento del tamaño de las instancias y combinaciones de recursos.

Los resultados permiten deducir que el algoritmo genético propuesto genera resultados iguales o mejores que el método exacto, no se presenta resultados sesgados o ajenos a la información de entrada.

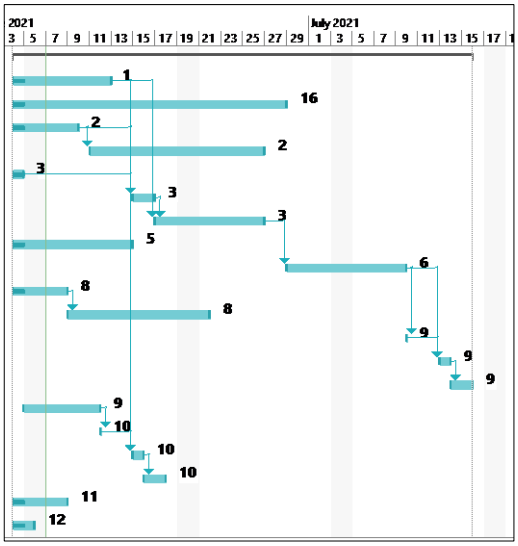
El tiempo promedio de solución para los diferentes escenarios usando el algoritmo genético propuesto, es ordenado de forma ascendente: $Zmin3$, $Zmin1$, $Zmin4$ y $Zmin2$. Es de resaltar que el conjunto de instancias del escenario $Zmin2$ era el que menos tiempo requería con el método exacto, con el algoritmo genético es el escenario que requiere más tiempo de solución, también es importante observar que la instancia $10w3m6r$ fue la que requirió más tiempo de solución con cualquiera de los dos métodos, puede justificarse en que la asignación aleatoria de recursos está más saturada en esta instancia.

5.7. RESULTADOS CASO REAL DE ESTUDIO

Finalmente, el AG es probado para solucionar un caso de estudio para un programa de producción real de la planta de inyección. El caso tiene la combinación de moldes, máquinas y tiempos necesarios para fabricar las piezas tipo “A” para un mes de producción. Estas piezas reciben esta calificación porque son las más costosas de producir en términos de recurso y requerimiento de materiales, también tienen gran volumen, aspecto que afecta su almacenamiento y WIP de la planta, son piezas críticas.

El caso consiste en secuenciar 25 trabajos en 15 máquinas utilizando 12 moldes disponibles, cinco de estos moldes tienen réplicas con menores tiempos de ciclo, para un total de 17 moldes. El “blend” de trabajos este compuesto por diferentes productos, con diferentes moldes, diferentes máquinas, diferentes ciclos de inyección y sus respectivas restricciones técnicas de montaje. Algunas piezas se pueden elaborar con el mismo molde pero esto no significa que los trabajos puedan ser realizados en la misma máquina, esta particularidad se presenta porque algunas piezas requieren un estándar mayor de calidad o estético que no se logra en todas las máquinas. El volumen del programa es de 333.700 piezas inyectadas. En la Figura 16, se muestra el Gantt usado en la compañía para programar la instancia real.

Figura 16. Diagrama de Gantt, programación actual en planta inyección.



Fuente: Autor.

En la Tabla 19, se presentan los valores en horas de $Zmin_1$, $Cmax$ y $T(j)$ para la programación de trabajos usando el método actual de la planta de inyección. τ es el tiempo en segundos necesario para que un programador elabore manualmente el programa de producción actual.

Tabla 19. Resultados Zmin de caso real, método actual.

Instance	δ_z^{ga}	$Cmax$	$T(j)$	τ
25w15m17r	1325,7	1008	2067	21600

Fuente: Autor.

El caso real con exactamente la misma configuración es solucionado con el algoritmo genético propuesto, los resultados en horas para $Zmin_1$, $Cmax$ y $T(j)$ se encuentran en la Tabla 20. τ es el tiempo de solución en segundos.

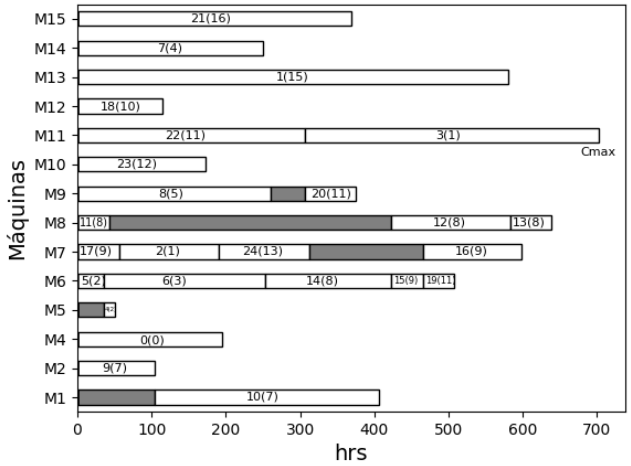
Tabla 20. Resultados caso real, usando algoritmo genético propuesto.

Instance	δ_z^{ga}	$Cmax$	$T(j)$	τ
25w15m17r	913,3	703	1404	25

Fuente: Autor.

En la Figura 17 se presenta el programa de secuenciación de los trabajos obtenido con el algoritmo genético propuesto, se observa que el algoritmo genético representa las características particulares del ambiente de producción de la planta de inyección, también asigna los recursos según las restricciones de compatibilidad que en algunos casos significan la asignación de un trabajo a solo un molde y maquina en específico.

Figura 17. Programa de producción con algoritmo genético.



Fuente: Autor.

En la Tabla 21, se presenta en términos porcentuales la variación en porcentaje de los componentes de la función objetivo embebida $Zmin_1 = \alpha * Cmax + \beta * T(j)$ y los componentes $Cmax$ y $T(j)$ respectivamente.

Tabla 21. Mejoramiento en porcentaje del programa de secuenciación usando GA.

$Componente$	Gap
$\alpha * Cmax + \beta * T(j)$	**-31,1%**
$Cmax$	**-30,2%**
$T(j)$	**-32,07%**

Fuente: Autor.

El algoritmo genético genera soluciones en promedio 30% mejores para $Zmin_1$, $Cmax\ y\ T(j)$, que el método usado actualmente en la planta de inyección. Como se mencionó en la Sección 5.3. el método exacto no puede solucionar instancias similares a un escenario de producción real, por su complejidad solo se solucionaron

instancias hipotéticas. *En cambio, el algoritmo genético propuesto si puede brindar soluciones de alta calidad para estas instancias, en tiempos razonables.*

En la Tabla 22 se presentan los ahorros en costos operacionales para la planta de inyección, como base del ejercicio de costos, se compara el $Cmax$ obtenido con el método actual de programación de la planta de inyección versus el $Cmax$ obtenido con el algoritmo genético propuesto. En el caso proyectado de realizar la programación de la producción con el algoritmo genético propuesto se ahorran 305 horas frente al método actual, para fabricar el conjunto de piezas *Tipo A*. El costo hora por tener habilitada una máquina de inyección, con su conjunto de periféricos, moldes y sistema de refrigeración es de $70.000 Cop/hora.

Tabla 22. Ahorros potenciales por secuenciar trabajos con el algoritmo genético.

$Método$	$Horas\ mes\ ahorradas$	$Ahorro\ Cop/mes$	$Ahorro\ en\ costo\ de\ operación/mes$
Algoritmo genético.	330	$21.000.000	30,2%

Fuente: Autor.

El ahorro podría ser mucho mayor si se programaran todas las piezas incluyendo las piezas tipo B y C, usando todo el conjunto de máquinas operativas, pero el caso de estudio de esta Sección es de las piezas tipo A y con un horizonte de tiempo de solo un mes.

La tardanza de los trabajos $T(j)$ baja 32% si se programa la secuenciación con el algoritmo genético propuesto, esta reducción ayuda a cumplir el objetivo fijado por gerencia de completar el programa de producción en el periodo estipulado para no afectar los demás eslabones de la cadena de producción, de esta forma también se asegura que se cumplan las ventas confirmadas con anterioridad a mayoristas, retailers y negocios B2B, de esta forma la compañía puede mantener su flujo de caja, evitar penalización por incumplimiento de entregas y conserva su posición en el nicho de mercado.

Otro beneficio de secuenciar los trabajos con el AG es que el equipo de programación tendrá 25% de sus jornadas laborales disponibles para actividades de gestión, este valor corresponde a 12 horas que invierten en realizar manualmente el programa de secuenciación actual. El ahorro en horas laborales corresponde a $360.000 mensuales, el valor de la mano de obra no es muy alto, pero es de resaltar que con este tiempo disponible (si se programa producción con el AG) el equipo de programación de producción estará enfocado en gestionar los recursos necesarios para garantizar que se ejecute el programa de producción.

6. CONCLUSIONES

El problema de secuenciación de trabajos en este proyecto se caracterizó usando un modelo matemático MILP, el conjunto de restricciones que se formularon permitieron representar correctamente en el modelo el entorno de producción real de la planta de inyección. El modelo se formuló pensando en realizar múltiples experimentaciones y poder probar diferentes escenarios con el fin de probarlo y usarlo como referente de comparación para otro método de solución.

Se propuso un algoritmo genético, donde se diseñó un cromosoma completamente original que represento el entorno de producción real de la planta de inyección, incluyendo las restricciones de elegibilidad y tiempos de alistamiento. También se desarrolló una metodología igualmente original para calcular el fitness del cromosoma dado que la función objetivo combinaba dos objetivos tradicionales ($Cmax$ y $T(j)$) buscando la minimización mutua.

Se crearon instancias de dos tipos: random y con información de producción real. Estas instancias fueron usadas para desarrollar los experimentos computacionales, comparar rendimiento de los métodos y validar resultados. Las instancias desarrolladas pueden ser usadas más adelante por otros investigadores para comparar y proponer mejores métodos de solución que el propuesto en este proyecto.

Respecto a los resultados de los experimentos computacionales, se puede afirmar que el algoritmo genético propuesto logra resultados iguales o mejores que el método exacto, en todos los tipos de instancias, es necesario recalcar que el algoritmo genético propuesto soluciona instancias reales que el método exacto no puede y crea una solución 1,18% mejor en promedio para todo conjunto de instancias random. El AG también crea una solución en promedio 30% mejor para secuenciar los trabajos de la planta de inyección, si se compara con el método usado actualmente. Los mejores resultados del algoritmo genético frente al método exacto se logran en las instancias de mayor tamaño y de mayor complejidad por tener más cantidad de combinaciones de recursos. Para lograr este rendimiento, el algoritmo genético requirió un ajuste de parámetros con diseño factorial, parámetros que se pueden aplicar universalmente a todo tipo de instancias y que aportó bastante a estabilizar el rendimiento del algoritmo. Fue interesante observar que $Cmax$ y $T(j)$ se comportaban como objetivos opuestos en algunos de los escenarios. Este comportamiento justificó el planteamiento de los cuatro escenarios y el planteamiento de la función objetivo embebida, sin significar que siempre se logren buenas soluciones.

Evaluando los tiempos de solución, el algoritmo genético propuesto, logra soluciones iguales o mejores para $Zmin, Cmax\ y\ T(j)$ que el método exacto en un tiempo promedio 97% menor. Al igual que muchas heurísticas existe un balance entre el tiempo de ejecución y la calidad de las soluciones, pero en el algoritmo genético propuesto no se aprecia una distorsión de la calidad de los resultados frente a los tiempos de solución, es más, se logran buenas soluciones en tiempos razonables. El Cmax del método actual de programación es de 42 días, el Cmax del AG es de 30 días, si comparamos los dos valores de Cmax, el algoritmo genético también sobresale, logrando reducir el Cmax del plan de producción para un conjunto de piezas tipo A en 12 días hábiles de producción.

El algoritmo genético propuesto fue desarrollado con software libre, su competitividad en términos de resultados y tiempos de ejecución exceden en beneficios si se hace una comparación con el método exacto, no solo en la calidad de las soluciones y el tiempo razonable para lograrlas sino también en un elemento muy importante, su costo de implementación es prácticamente gratis a diferencia del método exacto. El AG es más amigable para el trabajador con la elaboración del programa de producción, luego de que el trabajador se familiariza con la introducción de los datos, el proceso es menos tedioso que hacer un Gantt de forma prácticamente manual, como ocurre actualmente.

Si la programación de la producción se hace secuenciando los trabajos con el algoritmo genético propuesto, puede beneficiar a la planta de inyección reduciendo el tiempo invertido para desarrollar la secuenciación de los trabajos en aproximadamente dos jornadas laborales, pero los ahorros más importantes son la reducción de los costos operacionales que deja de generar la planta por extender la ejecución de su programa de producción más de lo necesario y el uso apropiado de sus recursos disponibles. Los ahorros proyectados pueden ascender a más de $200.000.000 de pesos anuales (puede ser mayor, esta cifra corresponde solo a programación de piezas tipo A), dinero que se puede reinvertir para hacer transformación tecnológica de la maquinaria y moldes de inyección con el objetivo de mejorar los indicadores globales de eficiencia operaciones e indicadores de cumplimiento de fabricación lotes de producción en cantidad y fecha.

Las perspectivas del proyecto a futuro deben ser la opción de programar el algoritmo en un lenguaje diferente a Python, a pesar de lograr buenos resultados se considera que un lenguaje más sofisticado puede generar mejores soluciones y es una necesidad si se desea realizar una implementación a nivel industrial. Para tener un ambiente de secuenciación de trabajos más dinámico, a futuro se deben plantear instancias que tengan parámetros con carácter probabilístico, se pueden considerar eventos como fallas de máquinas, fallos eléctricos, disponibilidad de recursos como puentes grúas o periféricos especializados y mantenimiento de recursos. De esta forma se pueden obtener buenas soluciones usando el algoritmo genético y poder

reducir la incertidumbre de un panorama de producción en muchas ocasiones incierto.

Como trabajo futuro también se deben considerar nuevos métodos de selección, mutación y crossover, los usados en el algoritmo genético propuesto fueron versátiles para encontrar buenas soluciones, pero a nivel computacional se pueden encontrar mejores soluciones con alternativas más robustas, es posible también crear un método compuesto soportando este algoritmo genético con una heurística ya probada. Este proyecto de grado se puede considerar como una primera aproximación sobre la que se puede trabajar para extender su aplicación a otro tipo de industrias donde los tiempos de trabajo están directamente relacionados con los recursos usados en su ejecución. El ajuste de parámetros es un componente primordial para un algoritmo genético, en futuras investigaciones se pueden desarrollar ajustes de parámetros más robustos para el AG basados en machine learning.

REFERENCIA BIBLIOGRAFICA

Acoplasticos. (2018). *Acoplasticos*. Messe Düsseldorf y Corferias firman alianza para fortalecer Colombiaplast. https://www.acoplasticos.org/index.php/mnu-noti/396-ns-210203.

Alcaldía de Medellín-CREAME. (2019). Fabricación de productos del plástico. https://empresarismo.medellindigital.gov.co/images/inteligencia_mercados/PDF/Fabricacion-de-productos-de-plastico.pdf

Adams, B. (1988). The shifting bottleneck procedure for job shop scheduling. Management Science, 34, 391–401.

Adulyasak, Y. (2019). An exact optimization approach for an integrated process configuration, lot-sizing, and scheduling problem. *Computers & Operations Research*, *103*, 310–323.

Anghinolfi, D., Paolucci, M., & Ronco, R. (2021). A bi-objective heuristic approach for green identical parallel machine scheduling. European Journal of Operational Research, 289 (2), 416–434. https://doi.org/10.1016/j.ejor.2020.07.020.

Afzalirad, M., & Rezaeian, J. (2016). Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. *Computers and Industrial Engineering*, *98*, 40–52. https://doi.org/10.1016/j.cie.2016.05.020

Avdeenko, T. V., Mezentsev, Y. A., & Estraykh, I. V. (2017). Heuristic Approach to Unrelated Parallel Machines Scheduling under Availability and Resource Constraints. *IFAC-PapersOnLine*, *50*(1), 13096–13101. https://doi.org/10.1016/j.ifacol.2017.08.1998

Bektur, G., Saraç, T. (2019). A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server.

Computers & Operations Research, 103. https://doi.org/10.1016/j.cor.2018.10.010

Bellman, R. (1955). Mathematical aspects of scheduling theory. Rand corporation.

Báez, S., Angel-Bello, F., Alvarez, A., & Melián-Batista, B. (2019). A hybrid metaheuristic algorithm for a parallel machine scheduling problem with dependent setup times. *Computers and Industrial Engineering*, *131*(March), 295–305. https://doi.org/10.1016/j.cie.2019.03.051

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization

metaheuristics. Information Sciences,, 237, 82–117.

Biskup, D., Herrmann, J., & Gupta, J. N. D. (2008). Scheduling identical parallel machines to minimize total tardiness. *International Journal of Production Economics*, *115*(1), 134–142. https://doi.org/10.1016/j.ijpe.2008.04.011

Brucker, P. (1998). *Scheduling: Theory and Applications*. 414–427. https://doi.org/10.1007/978-3-642-58891-4_66

Buchem, M., & Vredeveld, T. (2021). Performance analysis of fixed assignment policies for stochastic online scheduling on uniform parallel machines. *Computers and Operations Research*, *125*, 105093. https://doi.org/10.1016/j.cor.2020.105093

Cancela, H., Piñeyro, P., & Velázquez, J. (2018). A MILP formulation for a tire curing scheduling problem. *Electronic Notes in Discrete Mathematics*, *69*, 61–68. https://doi.org/10.1016/j.endm.2018.07.009

Cervantes-Sanmiguel, K. I., Vargas-Flores, M. J., & Ibarra-Rojas, O. J. (2020). A two-stage sequential approach for scheduling with lot-sizing decisions in the context of plastic injection systems. *Computers and Industrial Engineering*, *April*, 106969. https://doi.org/10.1016/j.cie.2020.106969

Chen, J. F., Wu, T. H. (2006). Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. Omega, 34, 81–89. https://doi.org/10.1016/j.omega.2004.07.023

Chergui, A., Hadj-Hamou, K., & Vignat, F. (2018). Production scheduling and nesting in additive manufacturing. *Computers and Industrial Engineering*, *126*(September), 292–301. https://doi.org/10.1016/j.cie.2018.09.048

Chung, T., Gupta, J. N. D., Zhao, H., & Werner, F. (2019). Minimizing the makespan on two identical parallel machines with mold constraints. *Computers and Operations Research*, *105*, 141–155. https://doi.org/10.1016/j.cor.2019.01.005

Dang, Q. V., van Diessen, T., Martagan, T., & Adan, I. (2020). A matheuristic for parallel machine scheduling with tool replacements. *European Journal of Operational Research*, *xxxx*. https://doi.org/10.1016/j.ejor.2020.09.050

Dastidar, S. G., & Rakesh, N. (2005). Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs. Computers & Operations Research, 32, 2987 – 3005.

Ding, J., Shen, L., Lü, Z., & Peng, B. (2019). Parallel machine scheduling with completion-time-based criteria and sequence-dependent deterioration. *Computers and Operations Research*, *103*, 35–45. https://doi.org/10.1016/j.cor.2018.10.016

Fang, K., Wang, S., Pinedo, M. L., Chen, L., & Chu, F. (2020). A combinatorial

Benders decomposition algorithm for parallel machine scheduling with working-time restrictions. *European Journal of Operational Research*, *xxxx*. https://doi.org/10.1016/j.ejor.2020.09.037

Fanjul-Peyro, L. (2020). Models and an exact method for the Unrelated Parallel Machine scheduling problem with setups and resources. Expert Systems with Applications, xxxx. https://doi.org/10.1016/j.eswax.2020.100022

Gantt, H. (1903). A Graphical Daily Balance in Manufacture. ASME.

Gerstl, E., & Mosheiov, G. (2012). Scheduling on parallel identical machines with job-rejection and position-dependent processing times. *Information Processing Letters*, *112*(19), 743–747. https://doi.org/10.1016/j.ipl.2012.06.009

Ghalami, L., & Grosu, D. (2019). Scheduling parallel identical machines to minimize makespan: A parallel approximation algorithm. *Journal of Parallel and Distributed Computing*, *133*, 221–231. https://doi.org/10.1016/j.jpdc.2018.05.008

Ghosh Dastidar, S., & Nagi, R. (2005). Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs. *Computers and Operations Research*, *32*(11), 2987–3005. https://doi.org/10.1016/j.cor.2004.04.012

Grefenstette, J. (1985). Proceedings of the 1st International Conference on Genetic Algorithms. In L. E. Associates (Ed.).

Gregory, R. W., & Muntermann, J. (2014). Heuristic theorizing: Proactively generating design theories. *Information Systems Research*, *25*(3), 639–653. https://doi.org/10.1287/isre.2014.0533

Holland, J. (1975). *Adaptation in Natural and Artificial Systems* (1st ed.). University of Michigan Press.

James, J. (1955). Notes on Some Scheduling Problems. University of California.

Jiang, X., Lee, K., & Pinedo, M. L. (2020). Ideal schedules in parallel machine settings. *European Journal of Operational Research*, *xxxx*, 1–13. https://doi.org/10.1016/j.ejor.2020.08.010

Richard Karp. (1972). In Complexity of computer computations (pp. 85-103). Springer.

Kılıç, H., & Yüzgeç, U. (2019). Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling. *Computers and Industrial Engineering*, *132*(June 2018), 166–186. https://doi.org/10.1016/j.cie.2019.04.029

Kramer, A., Dell'Amico, M., Feillet, D., & Iori, M. (2020). Scheduling jobs with release

dates on identical parallel machines by minimizing the total weighted completion time. *Computers and Operations Research*, *123*, 105018. https://doi.org/10.1016/j.cor.2020.105018

Kramer, A., Iori, M., & Lacomme, P. (2019). Mathematical formulations for scheduling jobs on identical parallel machines with family setup times and total weighted completion time minimization. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2019.07.006

Koza, J. (1994). Genetic programming as a means for programming computers by natural selection. Statistics and computing, 4, 87-112.

Li, Y., & Li, S. (2020). Scheduling jobs with sizes and delivery times on identical parallel batch machines. *Theoretical Computer Science*, *841*, 1–9. https://doi.org/10.1016/j.tcs.2020.06.023

Liu, M., & Liu, X. (2019). Satisfaction-driven bi-objective multi-skill workforce scheduling problem. *IFAC-PapersOnLine*, *52*(13), 229–234. https://doi.org/10.1016/j.ifacol.2019.11.134

Ma, R., Guo, S., & Miao, C. (2021). A semi-online algorithm and its competitive analysis for parallel-machine scheduling problem with rejection. *Applied Mathematics and Computation*, *392*. https://doi.org/10.1016/j.amc.2020.125670

Majumder, A., Laha, D., & Suganthan, P. N. (2018). A hybrid cuckoo search algorithm in parallel batch processing machines with unequal job ready times. *Computers and Industrial Engineering*, *124*(June), 65–76. https://doi.org/10.1016/j.cie.2018.07.001

Mařík, R. (2015). *Scheduling. FEE Czech Technical University.*

mincit. (2019). La industria manufacturera a enero de 2019. https://www.mincit.gov.co/getattachment/433a0476-f1ef-4a27-8af5-b2783c341509/Enero.aspx

Mönch, L., & Shen, L. (2021). Parallel machine scheduling with the total weighted delivery time performance measure in distributed manufacturing. *Computers and Operations Research*, *127*. https://doi.org/10.1016/j.cor.2020.105126

Montoya-Torres, J. R., Soto-Ferrari, M., & González-Solano, F. (2010). Production scheduling with sequence dependent setups and job release times. *(Prueba) DYNA (Prueba)*, *77*(163), 260–269.

Muter, I. (2020). Exact algorithms to minimize makespan on single and parallel batch processing machines. European Journal of Operations Research, 285 (2), 470–483. https://doi.org/10.1016/j.ejor.2020.01.065

Najat, A., Yuan, C., Gursel, S., & Tao, Y. (2019). Minimizing the number of tardy jobs on identical parallel machines subject to periodic maintenance. *Procedia Manufacturing*, *38*(2019), 1409–1416.

https://doi.org/10.1016/j.promfg.2020.01.147

Nguyen, N. Q., Yalaoui, F., Amodeo, L., Chehade, H., & Toggenburger, P. (2018). Total completion time minimization for machine scheduling problem under time windows constraints with jobs' linear processing rate function. *Computers and Operations Research*, *90*, 110–124. https://doi.org/10.1016/j.cor.2017.09.015

Nonås, S. L., & Olsen, K. A. (2005). Optimal and heuristic solutions for a scheduling problem arising in a foundry. *Computers and Operations Research*, *32*(9), 2351–2382. https://doi.org/10.1016/j.cor.2004.03.007

Ouazene, Y., & Yalaoui, F. (2018). Identical parallel machine scheduling with time-dependent processing times. *Theoretical Computer Science*, *721*, 70–77. https://doi.org/10.1016/j.tcs.2017.12.001

Oztemel, E., & Selam, A. A. (2017). Bees Algorithm for multi-mode, resource-constrained project scheduling in molding industry. *Computers and Industrial Engineering*, *112*, 187–196. https://doi.org/10.1016/j.cie.2017.08.012

Pinedo, M. (2016). Scheduling, theory, algorithms adn systems. Springer.

Ríos-Solís, Y., Ibarra-Rojas, O. J., Cabo, M., & Possani, E. (2020). A heuristic based on mathematical programming for a lot-sizing and scheduling problem in mold-injection production. *European Journal of Operational Research*, *284*(3), 861–873. https://doi.org/10.1016/j.ejor.2020.01.016

Şafak, C. U., Yilmaz, G., & Albey, E. (2019). A hierarchical approach for solving simultaneous lot sizing and scheduling problem with secondary resources. *IFAC-PapersOnLine*, *52*(13), 1931–1936. https://doi.org/10.1016/j.ifacol.2019.11.485

Sarac, T., Sipahioglu, A., & Akyol Ozer, E. (2017). A two-stage solution approach for plastic injection machines scheduling problem. *Journal of Industrial & Management Optimization*, *13*(5), 0–0. https://doi.org/10.3934/jimo.2020022

Sethanan, K., Wisittipanich, W., Wisittipanit, N., Nitisiri, K., & Moonsri, K. (2019). Integrating scheduling with optimal sublot for parallel machine with job splitting and dependent setup times. *Computers and Industrial Engineering*, *137*(September), 106095. https://doi.org/10.1016/j.cie.2019.106095

Shim, S. O., & Kim, Y. D. (2007). Scheduling on parallel identical machines to minimize total tardiness. *European Journal of Operational Research*, *177*(1), 135–146. https://doi.org/10.1016/j.ejor.2005.09.038

Silver, E. A. (2004). An overview of heuristic solution methods. *Journal of the Operational Research Society*, *55*(9), 936–956. https://doi.org/10.1057/palgrave.jors.2601758

Soares, L. C. R., & Carvalho, M. A. M. (2020). Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints. *European*

Journal of Operational Research, *285*(3), 955–964. https://doi.org/10.1016/j.ejor.2020.02.047

Soleimani, H., Ghaderi, H., Tsai, P. W., Zarbakhshnia, N., & Maleki, M. (2020). Scheduling of unrelated parallel machines considering sequence-related setup time, start time-dependent deterioration, position-dependent learning and power consumption minimization. *Journal of Cleaner Production*, *249*, 119428. https://doi.org/10.1016/j.jclepro.2019.119428

Soper, A. J., & Strusevich, V. A. (2019). Schedules with a single preemption on uniform parallel machines. *Discrete Applied Mathematics*, *261*, 332–343. https://doi.org/10.1016/j.dam.2018.03.007

Sorensen, K. S. (2017). A history of methaheuristics. Handbook of Heuristics , 791-808.

Tamura, Y. (2015). Studies on Advanced Metaheuristics employing Local Clustering for Job-shop Scheduling Problem.

Tanaev, V., Gordon, W., & Shafransky, Y. M. (2012). Scheduling Theory. Single-Stage Systems. Springer Science & Business Media.

Vincent, B., Duhamel, C., Ren, L., & Tchernev, N. (2016). An efficient heuristic for scheduling on identical parallel machines to minimize total tardiness. *IFAC-PapersOnLine*, *49*(12), 1737–1742. https://doi.org/10.1016/j.ifacol.2016.07.833

Wang, S., & Liu, M. (2015). Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. *Journal of Manufacturing Systems*, *37*, 182–192. https://doi.org/10.1016/j.jmsy.2015.07.002

Wu, L., & Wang, S. (2018). Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks. *International Journal of Production Economics*, *201*(February), 26–40. https://doi.org/10.1016/j.ijpe.2018.04.013

Yamada, T. (2003). Studies on Metaheuristics for Job shop and Flow shop Scheduling Problems.

Yang, T. (2009). An evolutionary simulation-optimization approach in solving parallel-machine scheduling problems - A case study. *Computers and Industrial Engineering*, *56*(3), 1126–1136. https://doi.org/10.1016/j.cie.2008.09.026

Yepes-Borrero, J. C., Perea, F., Ruiz, R., & Villa, F. (2020). Bi-objective parallel machine scheduling with additional resources during setups. *European Journal of Operational Research*, *xxxx*. https://doi.org/10.1016/j.ejor.2020.10.052

Zaidi, M., Jarboui, B., Kacem, I., & Loukil, T. (2010). Hybrid meta-heuristics for minimizing the total weighted completion time on uniform parallel machines. *Electronic Notes in Discrete Mathematics*, *36*(C), 543–550.

https://doi.org/10.1016/j.endm.2010.05.069

Zhang, X., Liu, S. C., Lin, W. C., & Wu, C. C. (2020). Parallel machine scheduling with linear deteriorating jobs and preventive maintenance activities under a potential machine disruption. *Computers and Industrial Engineering*, *145*(April), 106482. https://doi.org/10.1016/j.cie.2020.106482

Zheng, F., & Jin, K. (2019). An improved heuristic for single machine lot scheduling problem. *IFAC-PapersOnLine*, *52*(13), 217–222. https://doi.org/10.1016/j.ifacol.2019.11.115

Żurowski, M., & Gawiejnowicz, S. (2019). Scheduling preemptable position-dependent jobs on two parallel identical machines. *Computers and Industrial Engineering*, *132*(March), 373–384. https://doi.org/10.1016/j.cie.2019.03.043

ANEXO A. DISEÑO FACTORIAL MULTINIVEL PARA DEFINIR PARÁMETROS DE ALGORITMO GENÉTICO.

DEFINICIÓN Y AJUSTE DE PARÁMETROS USANDO DISEÑO FACTORIAL MULTINIVEL.

Los valores de los parámetros de población, crossover y mutación fueron seleccionados usando diseño factorial de tipo $5^1\ 3^2$, que considera un factor con cinco niveles y dos factores con tres niveles.

Para este diseño factorial son fijadas 45 corridas, por el carácter aleatorio del algoritmo genético se repitió cada experimento, es decir 90 corridas por cada instancia. El diseño factorial se desarrolló usando tres instancias como referencia, una pequeña y dos medianas. En total se realizaron 270 experimentos.

Los tratamientos para este diseño corresponden a 44 grados de libertad. Los factores y niveles establecidos para el diseño factorial se muestran en la Tabla 23.

Tabla 23. Factores y valores de niveles para diseño factorial.

Factor	Niveles				
Población	20	100	200	500	600
Crossover	0.125	0.25	0.5		
Mutación	0.05	0.1	0.15		

Fuente: Autor.

El diseño factorial se desarrolló con Minitab® 20.2, versión trial. El diseño factorial también se desarrolló en el software R, pero se optó por registrar en el documento los resultados de Minitab porque son más dicientes visualmente. Primero se definieron los factores, niveles y se generó el orden aleatorio para desarrollar el diseño factorial, obteniendo el valor de fitness para cada corrida (270 en total).

ANÁLISIS Y RESULTADOS

1. Análisis de varianza.

Con el análisis de varianza (Ver Figura 18) se puede observar que solo los factores population y crossover tienen un P-value significativo es decir menor a 0.05 con un nivel de confianza de 95%, por lo tanto, se rechaza la hipótesis nula para estos dos factores. También se rechaza la hipótesis nula para la interacción entre el factor

población y crossover, es decir la interacción entre estos dos factores aporta en gran parte a obtener un valor de fitness bajo.

Figura 18. Análisis de varianza, instancias de tamaño pequeño y mediano.

Instancia 5w5m6r

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	44	1,07600	0,02445	1,44	0,114
Linear	8	0,32000	0,04000	2,35	0,033
poblacion	4	0,19600	0,04900	2,88	0,033
crossover	2	0,09800	0,04900	2,88	0,066
mutacion	2	0,02600	0,01300	0,76	0,471
2-Way Interactions	20	0,54800	0,02740	1,61	0,092
poblacion*crossover	8	0,39200	0,04900	2,88	0,011
poblacion*mutacion	8	0,10400	0,01300	0,76	0,635
crossover*mutacion	4	0,05200	0,01300	0,76	0,554
3-Way Interactions	16	0,20800	0,01300	0,76	0,714
poblacion*crossover*mutacion	16	0,20800	0,01300	0,76	0,714
Error	45	0,76500	0,01700		
Total	89	1,84100			

Instancia 10w5m6r

Analysis of Variance

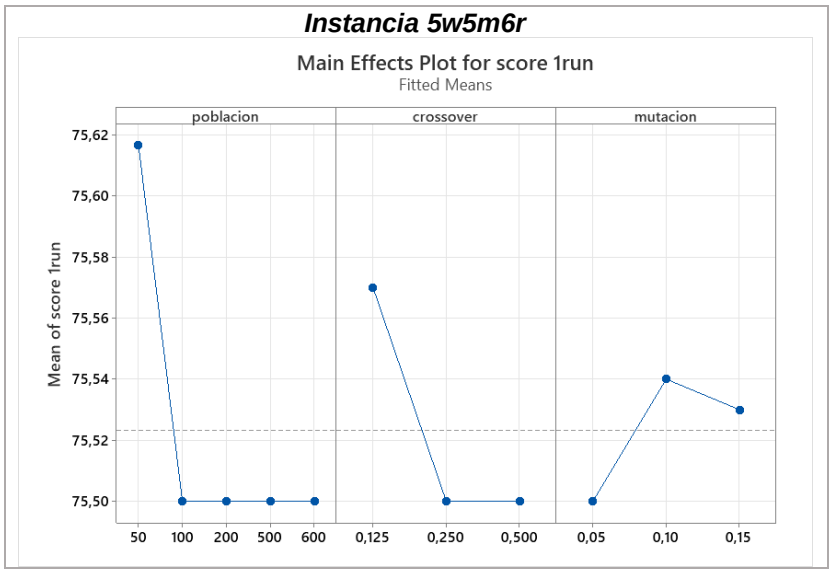
Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	44	9637,5	219,03	2,44	0,002
Linear	8	7623,2	952,90	10,61	0,000
poblacion	4	6799,4	1699,86	18,93	0,000
crossover	2	791,9	395,94	4,41	0,018
mutacion	2	31,9	15,95	0,18	0,838
2-Way Interactions	20	704,0	35,20	0,39	0,987
poblacion*crossover	8	545,2	68,15	0,76	0,640
poblacion*mutacion	8	63,7	7,96	0,09	0,999
crossover*mutacion	4	95,1	23,78	0,26	0,899
3-Way Interactions	16	1310,3	81,89	0,91	0,561
poblacion*crossover*mutacion	16	1310,3	81,89	0,91	0,561
Error	45	4040,1	89,78		
Total	89	13677,5			

Fuente: Autor.

2. Efectos principales de cada factor sobre el fitness e interacción entre factores.

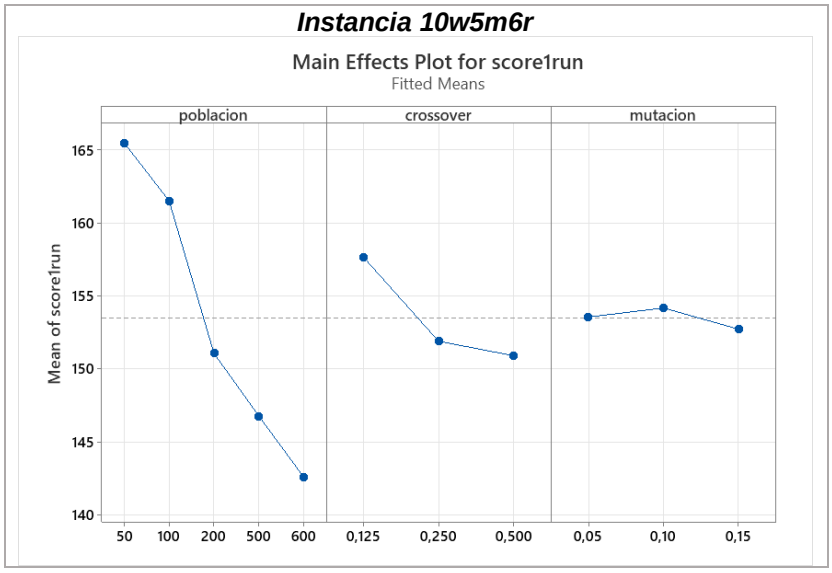
Se observa una relación directa entre valores altos de population y crossover con un menor valor de fitness (Ver Figura 19 y Figura 20), respecto a mutation es prudente un valor intermedio según los resultados.

Figura 19. Efecto de factores en el valor de fitness inst. 5w5m6r.



Fuente: Autor.

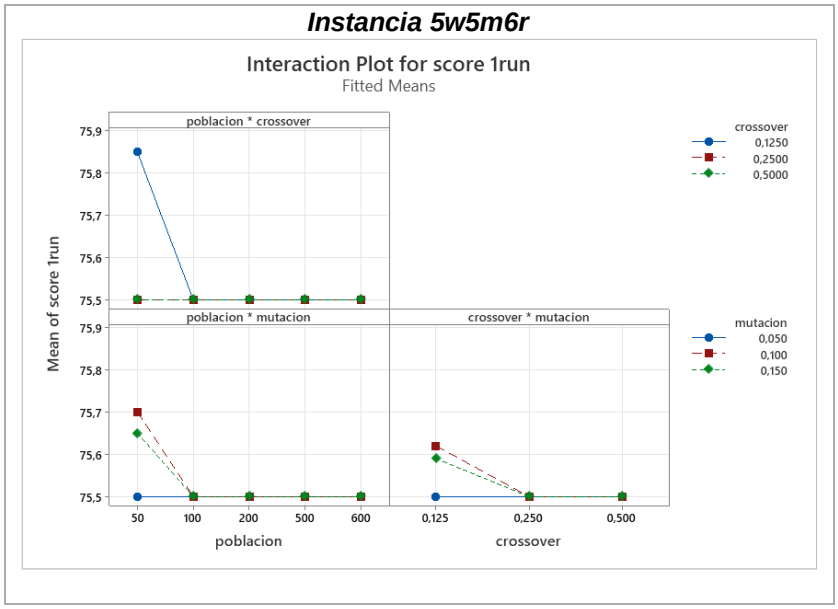
Figura 20. Efecto de factores en el valor de fitness inst. 10w5m6r.



Fuente: Autor.

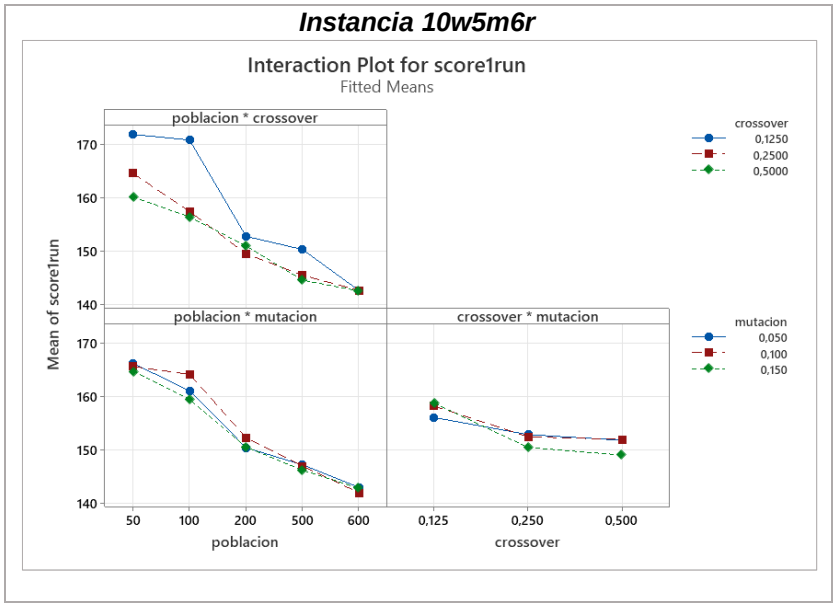
Respecto a la interacción entre factores y al cruce de sus proyecciones (Ver Figura 21 y Figura 22), se puede concluir que existe buena interacción entre los factores population/crossover y crossover/mutation para conseguir un nivel bajo de fitness.

Figura 21. Interacción entre factores y fitness, inst. 5w5m6r.



Fuente: Autor.

Figura 22. Interacción entre factores y fitness, inst. 10w5m6r.



Fuente: Autor.

3. DEFINICIÓN DE PARAMETROS.

Con el diseño factorial realizado se puede concluir que los valores más apropiados para los parámetros population, crossover y mutation que permiten el mínimo valor de fitness posible son los presentados en la Tabla 24 .

Tabla 24. Parámetros seleccionados para AG con diseño factorial.

Parámetro	Valor
Population	600
Crossover	0.5
Mutation	0.15

Fuente: Autor.

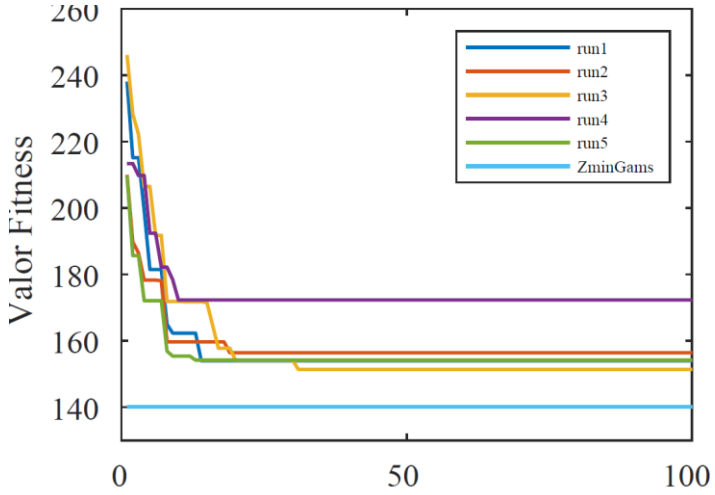
La selección de estos valores está justificada en los P-values, la interacción de los factores para minimizar el valor del fitness y las gráficas de interacción y efectos. La selección también está justificada en escoger los mejores valores con el fin de encontrar buenas soluciones para cualquier instancia en tiempo razonable.

4. SOLUCIÓN DE INSTANCIAS CON PARÁMETROS SELECCIONADOS.

La selección de parámetros se validó solucionando la instancia 10w5m6r, con parámetros sin ajustar (Figura 23) y con parámetros ajustados (Ver Figura 24).

Las instancias se solucionaron con el algoritmo genético propuesto en la Sección 4. Además, este procedimiento fue usado para definir la cantidad de generaciones que se deben producir con el objetivo de obtener un fitness cercano o igual al obtenido por el método exacto de solución, es decir 140,2 segundos.

Figura 23. Solución instancia 10w5m6r sin parámetros ajustados.



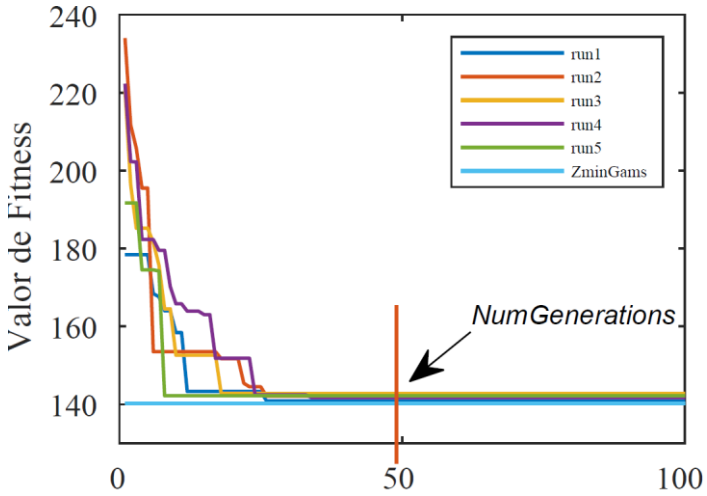
Fuente: Autor.

En la Figura 23, se muestra la solución de 5 corridas de la instancia 10w5m6r, se observa que el mejor fitness de cada corrida está lejos del Zmin del software Gams, en promedio existe un gap de 12,4% en este ejercicio.

En la Figura 24 se muestra la solución de la misma instancia usando los parámetros definidos con el diseño factorial. Se puede observar en la Figura 24 que los mejores valores de fitness en cada corrida están muy cercanos al $Zmin$ obtenidos con el método exacto, existe un gap promedio de 1,2% frente al método exacto de solución.

Con la solución de 10 instancias se puede observar convergencia en un “optima” durante la generación 50, por lo tanto, este será el valor fijado para este parámetro en todas las instancias. Con la experimentación computacional se observó que los parámetros seleccionados son universales y funcionan en instancias random y reales de todos los tamaños.

Figura 24.Solución instancia 10w5m6r con parámetros ajustados.



Fuente: Autor.

También se observó que en ocasiones el algoritmo converge en un valor de fitness que puede ser optimo o no, basado en las observaciones de todos los experimentos se define un criterio de stop luego de que se repite el valor de fitness por 15 generaciones, este valor le da la oportunidad al algoritmo de seguir generando soluciones sin parar antes de tiempo.

Tabla 25. Definición final de parámetros para algoritmo genético.

Parámetro	Valor
Population	600
Crossover	0.5
Mutation	0.15
Generaciones	50
Stop	15 repeticiones

Fuente: Autor.